APPROXIMATE AND ADAPTIVE ALGORITHMS FOR SOME OPTIMAL MOTION-PLANNING PROBLEMS

by

Hongyan Wang

Department of Computer Science
Duke University

Date: 10/25/96

Approved:

John Reif, Supervisor

Pankaj K. Agarwal

Ming Kao

Alan Biermann

Xin Zhou

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Computer Science
in the Graduate School of
Duke University

1996

ABSTRACT

(Computer Science)

APPROXIMATE AND ADAPTIVE ALGORITHMS FOR SOME OPTIMAL MOTION-PLANNING PROBLEMS

by

Hongyan Wang

Department of Computer Science
Duke University

Date: 10/25/96

Approved:

John Reif, Supervisor

Pankaj K. Agarwal

Ming Kao

Alan Biermann

Xin Zhou

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University

1996

Abstract

Robotic motion planning involves computing collision-free paths or trajectories for a robot moving amid obstacles. It is one of the most fundamental problems in robotics research. This dissertation presents three major results on some motion-planning problems.

Our first result is a novel approximation algorithm for the *kinodynamic motion-planning problem.* We consider the case of a point robot moving in a d-dimensional configuration space, with constrained dynamics. The constraints are given by upper-bounding the L_2 norms of the accelerations and velocities. Contrary to the previous approximation methods which use a uniform discretization in time space, our method employs a non-uniform discretization in configuration space (thus also a non-uniform one in time space). The discretization is coarser in regions that are farther away from any obstacles. The major consequence of such a discretization is a considerable decrease in the size of the search space, and in the running time, for many cases when the obstacles are sparse or are unevenly distributed.

The second result is a faster approximation algorithm for the 2D *curvature-constrained shortest-path problem.* In the problem, the robot is a point moving

in a plane whose path has a maximum curvature of 1, and the obstacles are polygons with a total of n vertices. The objective is to compute for the robot a shortest collision-free path that connects a given initial location and orientation to a final location and orientation. Compared to the previously best known result of Jacobs and Canny, the running time of our algorithm improves roughly from $O(n^3)$ to $O(n^2)$. More importantly, our running time does not depend on the total length of the obstacle edges, which can be arbitrarily large.

Our third result is an optimal algorithm for an on-line navigation problem, the *wall problem with penetrable obstacles*. In the original wall problem of Blum *et al.*, the point robot is to reach an oriented infinite line while moving amid oriented rectangular obstacles which are impenetrable. We generalize the obstacles to be penetrable and to have different weights. The objective is to minimize the weighted Euclidean distance.

Acknowledgements

This work would be impossible without the guidance, vision and encouragement from my advisor, Dr. John Reif. I learned from him the techniques as well as the attitude toward research. I am very thankful to Dr. Pankaj Agarwal, who helped me a lot with my writing and presentation. For one thing, I learned from him that technical papers should be not only accurate, but also artistic.

I would also like to thank my other committee members, Dr. Ming Kao, Dr. Alan Biermann and Dr. Xin Zhou, for their participation in my committee and their collaboration.

This work has been supported by NSF Grant NSF-IRI-91-00681, Rome Labs Contracts F30602-94-C-0037, ARPA/SISTO contracts N00014-91-J-1985, and N00014-92-C-0182 under subcontract KI-92-01-0182.

Thanks to Takemitsu Arakaki for his understanding, patience and his quiet support, which softened the hardness of life.

I thank all my friends and colleagues, especially Li-ling Huang, Wei Jin, Thomas Alexandra, Lei Tan, Song Liang, Qing Liu for their help and friendship.

I can never thank enough my parents, for their love, support and encouragement. Finally, I dedicate this dissertation to my late mother.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ultimate goal of robotics research is to develop *autonomous* robotic systems that can operate in an intelligent and independent way. An autonomous robot should have the three essential capabilities: *sensing, planning*, and *control*; see [95]. Sensing is the ability to gather information about the environment (through a variety of sensing devices such as tactile sensors, visual sensors, etc.), to analyze and to transform the raw data into a world model, based on which planning and control are executed. Planning ranges from high level *task planning* to low level *motion planning.* Task planning amounts to breaking up a complex task into a sequence of simple subgoals, whose combined execution will accomplish the desired task. This allows the robot to accept high level, general-term descriptions of activities. The control of a robotic system deals with how the actions are actually executed. The three components of sensing, planning, and control should interact with each other for a real autonomous system. For example, planning may direct the sensing system where and what information to gather, and new information from the sensing system may result

the planning component to re-evaluate its plan.

This dissertation studies some motion-planning problems. Motion planning is about computing paths or trajectories that can move a robot to desired places, without colliding the robot with the other components of the environment (e.g. equipment, wall, other robots). This problem is fundamental, since it is involved in almost all the tasks that we want a robot to perform, such as inspection, assembly, exploration, and reparation.

1.1 The Basic Motion-Planning Problem and Basic Concepts

The *basic motion planning problem*, known as the *Piano Movers' Problem* [91, 92, 93] is defined as follows. Let the robot be a collection of rigid subparts (some of which may be attached to each other by joints, while others may move independently), moving in a 2 or 3 dimensional Euclidean space filled with stationary objects called *obstacles*. Assume that the geometry and locations of both the robot and the obstacles are precisely known, and that there is no constraint on the motion of the robot, except the collision-free constraint and the constraints possibly imposed by joints. Given an initial placement I and a goal placement F, the problem is to find a collision-free path for the robot from I to F.

A *placement* of a robot is a specification of the position of every point of the robot (relative to a fixed reference frame). A placement is also referred to as a *configuration*. The set of all possible placements of the robot is the *configura-*

tion space of the robot. A robot is said to have k *degrees of freedom* (DOF) if its placement can be uniquely described by k real parameters. The configuration space for such a robot is $\mathbb{R}^k$. Here are some examples of configuration spaces for different systems. A point robot moving in a 3 dimensional Euclidean space has 3 DOF, and a 3 dimensional configuration space. A placement of a rod moving in a plane allowing translation and rotation can be specified by 3 parameters — 2 parameters specifying the position of one endpoint of the rod, and the third one the orientation of the rod. Thus the rod has 3 DOF, and its configuration space has 3 dimensions.

A placement is *free* if the robot with this placement does not intersect any obstacle. The set of all free placements is called the *free configuration space*, while its complementary set is called the *configuration obstacle*.

The *geometric complexity* of a motion-planning problem refers to the number of polynomial constraints used to define the configuration obstacle. The maximum degree of these polynomials is called *algebraic complexity* of the problem. The complexity of a motion-planning problem, or a motion-planning algorithm is usually described as a function of the dimension of the configuration space (DOF), the geometric complexity, and the algebraic complexity of the problem.

1.2 Classification of Motion-Planning Problems

There are far more varieties of motion-planning problems which can be encountered in reality than what is included in the basic motion-planning problem.

The problem space of robotic motion planning can be viewed to have the following dimensions:

- *Dimension of configuration space*: The dimension of a configuration space is determined by DOF, which can range from very small (2 for a point robot in the plane), to very large for a robot with many joints, or for a system with many independently moving robots.

- *Dynamic constraints*: In most of the robotic literature, the robot is assumed to be a *free flying* object that can follow any type of paths. This is not true for most real robotic systems that are subject to the so-called dynamic constraints. Some dynamic constraints include bounds on accelerations and velocities, or bounds on curvatures. These constraints are due to, for example, limited force or torque from motors.

- *Incomplete information*: We can not make the assumption that the robot has a complete and accurate map of the environment, in cases when the environment is dynamically changing, or the environment is uncharted (and the robot's mission is to explore it). The problem of motion planning without a prior knowledge is called the *on-line motion-planning problem.*

- *Uncertainty in control*: In the ideal situation, a robot is assumed to have a perfect control that allows it to follow the geometrical paths generated by the motion planner exactly. However, errors do occur in a robot's control and may result in diversion from the original path and collision with obstacles. Motion-planning problems that take into account the uncertainty in control are called *fine motion-planning problems.*

- *Moving obstacles*: In the basic motion planning problem, the obstacles are assumed to be stationary. If we consider the case when some (or all) of the obstacles are moving (and their motions are known in advance), the resulting problem is called the *dynamic motion-planning problem.* Dynamic motion planning arises in applications like planning for a robot arm in a workspace with various machines parts making predictable movements, or planning for a spacecraft navigating among a field of asteroids with known trajectories.

- *Movable objects*: Movable objects are like stationary obstacles, except that their placements can be changed by a robot. The most usual way for a robot to change the placement of a movable object is to grasp it or to push it. The presence of movable objects has a considerable impact on motion planning. For example, even when there exists no collision-free path between two placements in a given arrangement of the workspace, the robot may find one by displacing some movable objects.

- *Optimal motion planning*: We can associate cost functions with paths. Some of the cost functions may be path length, time length (for trajectories), number of corners and clearance (i.e., the distance to the closest obstacles). Sometimes we desire paths with optimal costs, and the problem of finding such paths is called the *optimal motion planning* problem.

The complexity result of the basic motion-planning problem is rather negative. If the degree of freedom is a part of the input, the problem is known to be PSPACE-hard [16, 85]. Canny [16] showed that it is PSPACE-complete by giv-

ing an algorithm whose running time is exponential in the degree of freedom. In light of the complexity, research has been focused on developing efficient algorithms for special cases, involving small degrees of freedom (see [62, 95] and references therein). For systems with many degrees of freedom, practical methods are restricted to heuristics using *potential field methods* [37, 38, 89], *randomization* [7, 55, 56], and *learning* [55, 56, 79].

The complexity of motion-planning problems grows fast as one adds dimensions to the basic motion planning problem. For example, planning the motion of a point robot in the plane, with bounded velocity modulus, among convex polygonal obstacles moving at constant linear velocity without rotation, is shown to be NP-hard in the number of obstacles [15]. Planning for a point robot in presence of uncertainty in 3 dimensional space with polyhedrals is Nondeterministic-Exponential-Time-Hard in the number of polyhedral faces [19].

In the rest of the review, we will focus our attention to optimal motion planning and planning with dynamic constraints.

1.3 Optimal Motion Planning

One of the extensively studied optimal motion-planning problems is the *unconstrained shortest-path* problem in 2D. The problem is to compute, for a point robot, the shortest path between two given points, such that the path does not intersect the interior of a given set of polygons (considered as obstacles). The algorithms developed for this problem can be classified as the *visibility-graph*

method, and the *shortest-path-map* method.

The visibility-graph method is based on the observation that a shortest path is a sequence of line segments, whose endpoints are obstacle vertices (we consider the start point and the goal point as vertices also). The basic idea is to build a weighted graph G as follows. The graph nodes are the obstacle vertices. There is an edge between two nodes if and only if the two vertices are visible from each other (i.e., the straight line segment connecting these two vertices does not intersect the interior of obstacles); and the weight of the edge is the Euclidean distance between these two vertices. Now the problem is turned into finding the shortest graph-path in G, which can be done using any Dijkstra-type algorithms [3]. The critical step is to construct the visibility graph efficiently. This direction is pursued by [4, 53, 94, 80, 86, 101], culminating in an optimal $O(n \log n + |E|)$-time algorithm by Ghosh and Mount [40], where n is the number of obstacle vertices, and $|E|$ is the number of graph edges. Unfortunately, the visibility graph can have $\Omega(n^2)$ edges in the worst case, so any shortest path algorithm that depends on an explicit construction of the visibility graph will have a similar worst-case running time of $\Omega(n^2)$.

The shortest-path-map method is to build a planar subdivision of the plane with respect to the start point s, so that all points in the same region of the subdivision have the same vertex sequence in their shortest paths from s. The map can also be used to answer single-source shortest path queries. This approach was taken by [46, 72, 73], where in [46], Hershberg and Suri gave an optimal $O(n \log n)$-time algorithm (the algorithm is optimal in the sense that it matches the lower bound on running time).

There has been work on computing shortest paths inside a simple polygon [41, 45, 66], and computing *rectilinear* shortest paths [28] (where the path length is taken in L_1 norm). Mitchell and Papadimitriou [76] studied the problem of finding a weighted shortest path through a planar subdivision in which each region has an associated weight. The running time of their algorithm is $O(n^7 L)$, where n is the number of edges in the subdivision and L is the precision of weights. The robot can be some geometric objects other than a point. Chew [21] gave an $O(n^2 \log n)$ algorithm for computing a shortest path for a disc, and Hershberger and Guibas [44] gave an $O(n^2)$ algorithm for a non-rotating convex body.

The shortest-path problem in $3D$ is substantially more difficult than the 2D one; in fact, it was proved to be NP-hard by Canny and Reif [19]. Even though the shortest path is still a sequence of line segments, each segment can have its endpoints any where in the interior of an edge. This is why the problem can not be transformed to a discrete search problem. An exact, exponential-time algorithm was given in [86], by transforming the problem to a decision problem of the first-order theory of reals. There are some special cases admitting polynomial-time algorithms. One such case is to calculate the shortest paths along the surface of a single convex polyhedron [74, 96], where [74] gave an $O(n^2 \log n)$-time (n being the number of vertices) algorithm for this problem. In view of the complexity of the 3D shortest-path problem, it is only practical to develop *approximation algorithms*, which is introduced in Section 1.4.

There are other optimal motion planning problems. A review of the on-line

motion planning problems is given in Section 1.6.3, while previous work on optimal motion-planning problems with dynamic constraints is introduced in Section 1.6.1 and Section 1.6.2.

1.4 Approximation Algorithms

Many of the optimal motion planning problems are intractable (i.e. they can not be solved by polynomial-time algorithms). For example, the unconstrainted shortest-path problem in 3D is NP-hard. We must consider approximation algorithms for these problems for practical reasons.

An algorithm is an *approximation algorithm*, if for any $\varepsilon > 0$, the algorithm can compute a solution whose cost (e.g., path length in the above mentioned example) is at most $(1+\varepsilon)$ times the optimal cost. An approximation algorithm is called *fully polynomial* if its running time is polynomial both in the input parameters of the problem, and in $(1/\varepsilon)$.

A fully polynomial approximation algorithm for the 3D shortest-path problem was developed by Papadimitriou [82]. The basic idea is to break an obstacle edge into a number of short segments, such that each segment can be approximated by its middle point. In this way, the problem is turned into one of building a visibility graph among the obstacle vertices and the middle points. The problem was later studied by Choi *et al.* [22, 23], who developed approximation algorithms in the *bit framework*, where the running time of the algorithm also depends on the number of bits to represent any algebraic values.[1]

[1]Most algorithms are developed within the context of one of two distinct computational frameworks, the *algebraic framework* and the *bit framework*. In the algebraic framework, each algebraic operation takes $O(1)$ time and returns an exact value. The complexity of an

Even for problems with polynomial-time exact solutions, it is desirable to consider approximation algorithms that run faster than the exact ones, in cases when approximation solutions are good enough and we care more about getting a solution fast. For example, for the 2D shortest-path problem, Clarkson [24] developed an approximation algorithm. The algorithm preprocesses in time $O(n \log(n/\varepsilon))$ to build a data structure of size $O(n/\varepsilon)$. After this, it can answer a query for a near optimal path from point s to point t in time $O(n/\varepsilon + n \log n)$.[2] The algorithm approximates the visibility graph by a sparser graph of a linear number of edges. The sparse graph satisfies the property that if there is an edge between two nodes in the original visibility graph, then there is a path (with possibly more than one edge) connecting the two nodes in the sparse graph and the length of the path is within $(1 + \varepsilon)$ factor of the length of the edge.

A review on approximation algorithms for motion planning with dynamic constraints will be given in Section 1.6.

1.5 Dynamic Constraints

A robot with a dynamic constraint cannot translate or rotate freely in the workspace. The dynamic constraints are classified as *holonomic* and *nonholo-*

---

algorithm in the algebraic framework depends only on the number of algebraic operations needed. In the bit framework, algebraic values are represented by binary strings. Thus each operation's time depends on the length of the binary strings representing its inputs, and the operation may not return an exact value. Most of the algorithms described so far are in the algebraic model.

[2]The algorithm was developed before the optimal $O(n \log n)$-time algorithm was found. It is worth mentioning here, since the algorithm is much simpler, thus more practical than the optimal exact one.

nomic. A *holonomic constraint* is one that can be expressed as an equation relating *only* configuration parameters. The consequence of a holonomic constraint is usually that it forbids the robot to be within a subset of its configuration space. This type of constraints does not incur special treatment, since they can be transformed to and treated as configuration obstacles. Typical holonomic constraints are those imposed by the prismatic and revolute joints of a manipulator arm.

Let $\mathbf{q}$ be a vector of configuration parameters of a robot, and $\dot{\mathbf{q}}$ be its first derivative with respect to time t. A constraint of one of the following forms

$$G(\mathbf{q}, \dot{\mathbf{q}}, t) < 0 \text{ or } G(\mathbf{q}, \dot{\mathbf{q}}, t) \geq 0 \tag{1.1}$$

is a *nonholonomic constraint* if G is a smooth *non-integrable* function, i.e., $\dot{\mathbf{q}}$ can not be eliminated from the equations. Unlike the holonomic constraints, a nonholonomic constraint expresses relations not only involving configuration parameters, but also involving their derivatives.

Let's take a look at the dynamic constraints of a car-like robot. A *car-like robot* is a four-wheel front-wheel-drive vehicle moving on a flat ground (see [62] for a more detailed description). The robot is modeled as a rectangle moving on a plane. Its placement is represented by a triple (x, y, θ), where $(x, y) \in \mathbb{R}^2$ are the coordinates of the midpoint R between the two rear wheels, and $\theta \in [0, 2\pi)$ is the angle between the main axis of the car and the x-axis; see Figure 1.1. The motion of a car-like robot is constrained so that the curve γ traced out by the mid-point R between the two rear wheels must be tangent to the main axis of the car at any moment. This implies that the car can not move sideways.

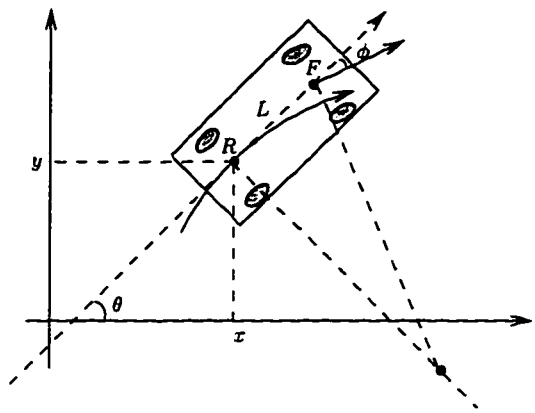

Figure 1.1: A car-like robot obeying nonholonomic constraints.

This constraint can be expressed as

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0. \tag{1.2}$$

The change in the car's orientation is determined by the *steering angle* ϕ, which is the angle between the main axis of the car and the direction of the mid-point F between the two front wheels. In general, mechanical stops in the steering gear constrain ϕ to be such that $|\phi| \leq \phi_{\max} < \pi/2$. The implication of this constraint is that the curve followed by R has a curvature (wherever it is defined) upper-bounded by

$$\frac{1}{\rho_{\min}} = \frac{\tan\phi_{\max}}{L}, \tag{1.3}$$

where L is the distance between F and R. $\rho_{\min} = L/\tan\phi_{\max}$ is called the *minimum turning radius* of the car. Rewriting this constraint in the forms of (1.1), we obtain

$$\dot{x}^2 + \dot{y}^2 - \rho_{\min}^2\dot{\theta}^2 \geq 0. \tag{1.4}$$

The two constraints of (1.2) and (1.4) are shown to be nonholonomic.

Some of the nonholonomic systems that have been studied include: a mobile robot with a maximumly achievable acceleration and a maximumly achievable velocity [17], a car-like robot [65], a system of multi-fingered hands rolling on the surface of a grasped object [68], space robots obeying the conservation of total angular momentum [100], etc.

The nonholonomic constraints raise the following main issues:

- How to decide whether a constraint is nonholonomic, i.e., how to determine whether the derivatives $\dot{\mathbf{q}}$ can be eliminated from the function expressing the constraint. This problem is answered by Frobenius Integrability Theorem [98].

- How to decide whether a nonholonomic constraint restricts the set of configurations reachable within a connected component of the free configuration space. This concerns the *controllability* of a robot with nonholonomic constraints. A nonholonomic constraint, unlike a holonomic one, can not be reduced to configuration obstacles. But it may complicate the reachability within a connected component of the free configuration space. A robot is *fully controllable* if and only if, if there exists a free path between any two configurations, then there also exists a free path obeying nonholonomic constraints between these two configurations. The controllability issue was studied in [63, 65, 67], where [63] showed the full controllability of a car-like robot. A tractor-trailer was proven to be fully controllable in [8, 67].

- How to generate paths subject to the nonholonomic constraints. With

nonholonomic constraints, it is not enough to describe a robot of k degrees of freedom with k parameters. We also need to specify derivatives of the k parameters. The conventional *configuration-space technique* does not apply here. The basic of this technique is to compute paths in the configuration space instead of in the workspace. It works for the unconstrained basic motion planning problem, because it is true that there exists a collision-free path between an initial placement and a final placement in the workspace if and only if the two placements lie within the same connected component of the free configuration space. This is not necessarily true if the robot has to obey nonholonomic constraints, since the paths obtained by the configuration-space technique may violate the nonholonomic constraints.

The third issue is the focus of this dissertation.

1.6 Summary of Previous Work and Our Results

1.6.1 Kinodynamic motion planning

The kinodynamic motion planning studies the problem of computing collision-free, minimum-time trajectories for a robot whose motion is governed by Newtonian dynamics and whose accelerations and velocities are bounded. A *trajectory* is a map $\Gamma : [0, T] \rightarrow \mathbb{R}^d \times \mathbb{R}^d$ given by $\Gamma(t) = (p(t), \dot{p}(t))$, where $p(t)$ and $\dot{p}(t)$ specify the location and the velocity of the robot at time t respectively, in a d dimensional configuration space. $\ddot{p}(t)$ is the acceleration function, which

determines a trajectory uniquely, once an initial state is fixed. The dynamic constraints are given by bounding the norms of the accelerations and velocities. The most studied norms are the L_∞ norm (called the *decoupled* case) and the L_2 norm (called the *coupled* case). The decouple case is simpler than the coupled one, because each dimension is independent of another, and a d-dimensional problem can be reduced to a 1-dimensional one. We require that $\|\dot{p}(t)\|_2 \leq 1$, and $\|\ddot{p}(t)\|_2 \leq 1$, for $0 \leq t \leq T$.[3] This problem bears major significance in robotic engineering. For example, a robot arm has to move not only in a collision-free fashion, but also in conformation of the dynamic bounds due to limited force or torque from motors.

With the exceptions of one and two dimensional cases (see Ó'Dúnlaing [78] and Canny *et al.* [18]), there are no exact solutions for the kinodyanmic motion planning problem. In fact, as an implication of the result given by Canny and Reif [19], the problem is at least NP-hard in 3 and higher dimensions. In light of this lower bound, most study has focused on finding approximate solutions. The earlier approximation algorithms of Sahar and Hollerbach [90] did not guarantee goodness of their solutions. Moreover, the running time was exponential in resolution. Canny *et al.* [17, 35] developed the first provably-good, polynomial-time approximation algorithm for the decoupled kinodynamic case. Their work was followed up by a series of work, in which Donald and Xavier [31, 33] improved the running time for the decoupled case, Heinzinger *et al.* [43], and Donald and Xavier [32, 34] inves-

[3] In previous literature, $\|\dot{p}(t)\|_2 \leq v_{\max}$, and $\|\ddot{p}(t)\|_2 \leq a_{\max}$, where $v_{\max} > 0$ and $a_{\max} > 0$ are arbitrary. However, we can always scale $v_{\max}$ and $a_{\max}$ to 1, by scaling time and the size of the configuration space; a proof can been seen in Section 2.8.

tigated the problem for open chain manipulators, and independent work of Donald and Xavier [30, 32, 34], and Reif and Tate [88] gave approximation algorithms for the coupled kinodynamic problem. The best known result for the coupled case, given by [34], is that given any $\varepsilon > 0$, one can find in time $O(c(d)p(n,\varepsilon,d)L^d(1/\varepsilon)^{6d-1})$, an approximate trajectory whose time length is at most $(1+\varepsilon)$ times the time length of an optimal trajectory, where d is the dimension, L is the size of the configuration space where the robot is confined to, n is the number of constraints describing the configuration obstacle, $c(d)$ is a function depending soly on d, and $p(n,\varepsilon,d)$ is a low-order polynomial in n,ε and d.

Even this time bound is very large in terms of $1/\varepsilon$. This results from a uniform discretization of the time space, which gives a large search space. To improve, our approximation algorithm for the coupled kinodynamic problem presented in Chapter 2 employs a non-uniform discretization of the configuration space. The discretization has the property that it is coarser in regions that are farther away from any obstacles. The major consequence is a considerable decrease in the size of the search space. (Non-uniform discretization is often used in solving PDEs, but focusing on different issues (see Miller *et al.* [69, 70] and references therein). Applying the idea to kinodynamic motion planning was first suggested by Xavier [103] but without rigorous proof. As it happens, provably good bounds given by us are quite intriguing to obtain.) The discretization is also such that we can obtain a set of canonical trajectories, which once computed, can be used repeatedly for different obstacle arrangements. Our algorithm can compute a trajectory whose time length is no more

than $(1+\varepsilon)$ times the time length of an optimal $3l$-safe trajectory (roughly speaking, a trajectory is safe if its path lies within a tube of free configuration space), in time $O(nN + N \log N(1/\varepsilon)^{4d-2})$, where N is bounded by $O((L/l)^d)$, but can be much smaller for cases when the obstacles are sparse, or are unevenly distributed. Compared to the previously best result of [34], the running time of our algorithm decreases in terms of the exponent of $1/\varepsilon$, from $6d-1$ to $4d-2$ (Notice that the $O((1/\varepsilon)^{3d})$ result of [33] is for the decoupled case). Also, our method is able to take advantage of the obstacle arrangements. As an application of the techniques developed for the kinodynamic motion planning, we are able to give the first known polynomial-time approximation algorithm for the curvature-constrained shortest path problem in 3 and higher dimensions (see Section 1.6.2 for the definition of the curvature-constrained shortest-path problem and previous work on it).

Our non-uniform discretization is based on a box decomposition of the configuration space. Other geometric algorithms using similar box decompositions include the work of Mitchell *et al.* [75] on ray-shooting problems and that of Hershberger and Suri [46] on 2D unconstrained shortest-path problem. Using non-uniform discretization for geometric planning with nonholonomic constraints presents new challenge.

Instead of using a deterministic discretization, Kavraki *et al.* [55, 56] developed a *random sampling technique*, where in preprocessing, grid points (called *milestones*) are chosen *randomly* and are connected by feasible paths to form a network. Then the network is used to answer planning queries. The method is similar to ours in that both do a preprocessing to obtain a set of valid paths

(or trajectories). However, they have to preprocess for each new environment, where in our method, the preprocessing is only done once for some fixed parameters, and can be used repeatedly for different environments.

1.6.2 Curvature-constrained shortest paths

Let $P : I \rightarrow \mathbb{R}^d$ be a d dimensional continuous differentiable path parameterized by arc length $s \in I$. The *average curvature* of P in the interval $[s_1, s_2] \subseteq I$ is defined by $\|\dot{P}(s_1) - \dot{P}(s_2)\|/|s_1 - s_2|$. In the curvature-constrained shortest-path problem, we require that the path of the point robot have an average curvature at most 1 in every interval.[4] In 2D, this restriction corresponds naturally to constraints imposed by a steering mechanism on a car-like robot (see Section 1.5). Figure 1.2 compares a curvature-constrained shortest path (drawn in solid line) and an unconstrained shortest path (drawn in dashed line) in 2D. The two paths start and end at the same locations. Since curvature is about how fast the orientation is changing, not only the location of the robot, but also its orientation has to be taken into account. The initial and final orientations are represented by arrows in Figure 1.2. It happens that the unconstrained shortest paths have "corners" at the obstacle vertices (for example, vertices a and b in Figure 1.2). The curvatures at these corners are infinite. Thus this type of paths can not be followed by a robot with a bounded maximum curvature.

[4]The reasons that we use average curvature instead of curvature are: 1) The curvature of a differentiable path may not exist at certain points; 2) If we consider the class of paths that have a curvature everywhere, the minimum-length path may not belong to this class; 4) The curvature of a path (wherever it is defined) is bounded by a number if and only if its average curvature is bounded by the same number; 3) The set of paths with bounded average curvatures sufficiently models paths in practice.

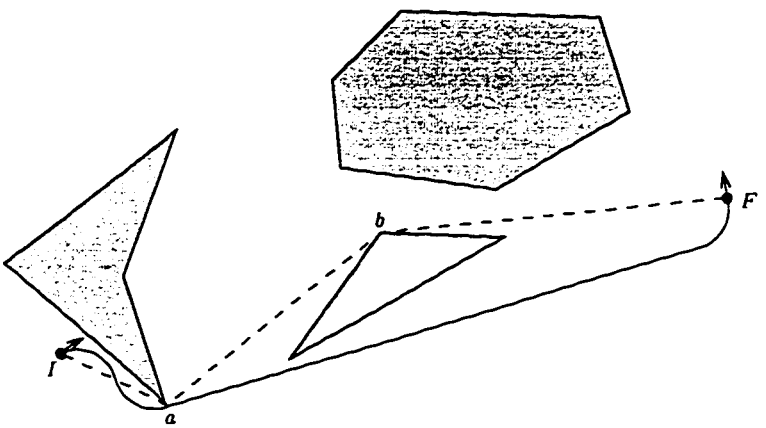

Figure 1.2: A comparision between a curvature-constrained shortest path and an unconstrained shortest path.

Dubins [36] was perhaps the first to study the curvature-constrained shortest paths. He proved that, in absence of obstacles, a curvature-constrained shortest path from any start position to any final position consists of at most 3 segments, each of which is either a straight line or an arc of a unit-radius circle. Reeds and Shepp [84] extended this obstacle-free characterization to robots that are allowed to make reversals. Using ideas from control theory, Boissonnat *et al.* [13] gave an alternative proof for both cases, and recently Sussmann [99] was able to extend the characterization for the 3-dimensional case. In presence of obstacles, Fortune and Wilfong [39] gave a $2^{poly(n,m)}$-time algorithm, where n is the total number of vertices in the polygons defining the obstacles, and m the number of bits of precision with which all points are specified; their algorithm only decides whether a path is feasible, without necessarily finding one. Jacobs and Canny [14, 48] gave an $O((\frac{n+L}{\epsilon^2})^2 + n^2(\frac{n+L}{\epsilon^2})\log n)$-time algorithm that finds an approximate path whose length is no more than $(1+\epsilon)$ times the

length of a shortest ε-*robust* path, where L is the total edge length of the obstacles. (Informally, a path is ε-robust if perturbations of certain points along the path by $\pm\varepsilon/2$ — in distance or in angle — do not violate the feasibility of the path.) The path returned by this algorithm is not necessarily robust. They also presented an $O(n^4 \log n + (\frac{n+L}{\varepsilon^2})^2)$-time algorithm that computes an $(\varepsilon/2)$-robust path whose length is no more than $(1+\varepsilon)$-times the length of an optimal ε-robust path. For the restricted case of *moderate obstacles*, i.e., when the curvature of the boundary of obstacles is also bounded by 1, Agarwal *et al.*[2] give efficient approximation algorithms, and Boissonnat and Lazard [12] give the first known polynomial-time algorithms for computing the exact shortest paths for the case when the edges of obstacles are circular arcs of unit radius. Wilfong [102] studies a restricted problem in which the robot must stay on one of m line segments (thought of as "lanes"), except to turn between lanes. For a scene with n obstacle vertices, his algorithm preprocesses the scene in time $O(m^2(n^2 + \log m))$, following which queries are answered in time $O(m^2)$. There has also been work on computing curvature-constrained paths when B is allowed to make reversals [8, 60, 61, 64, 71].

In Chapter 3, we present faster approximation algorithms for the 2D curvature-constrained shortest-path problem. Compared to the previously best known results of Jacobs and Canny [48], the running time of our algorithm not only improves in terms of the obstacle complexity, (if n is the number of obstacle vertices, our running time is $O(n^2)$ instead of $O(n^3)$), but more importantly it does not depend on L, which is the total length of obstacle edges. This factor can be large and can not be scaled down due to the curvature constraint. A

part of the improvement comes from a stronger characterization of curvature-constrained shortest paths in presence of obstacles, which is interesting in its own right.

1.6.3 On-line navigation

On-line motion planning arises when the robot has to navigate in an environment of which it has no *a priori* knowledge. The robot can only learn partial information (through various sensors) as it navigates. The objective of an on-line problem is usually to minimize the Euclidean distance traveled by the robot to finish its task. Because the robot makes plans based on partial information, it is not possible that any on-line algorithm can guarantee to return optimal paths. Because of this, *competitive ratio* (introduced by Sleator and Tarjan [97]) is used as a measure of goodness for on-line algorithms. The competitive ratio of an on-line algorithm is defined to be the *worst case* ratio of the performance (e.g., Euclidean distance) of the algorithm to an optimal off-line performance (i.e., when all the information is known). An algorithm is *optimal* if its competitive ratio matches the lower bound on the competitive ratios of the problem.

Blum *et al.* [11] studied the following on-line navigation problem, the *wall problem.* In this problem, the robot is a point moving on a plane, while the obstacles are oriented rectangles. The robot is to reach an oriented infinite line distance n away from its starting point. Even though the scenario is simple, the problem is quite interesting, because it embodies many of the key issues of on-line navigation. Blum *et al.*gave an optimal on-line algorithm for this

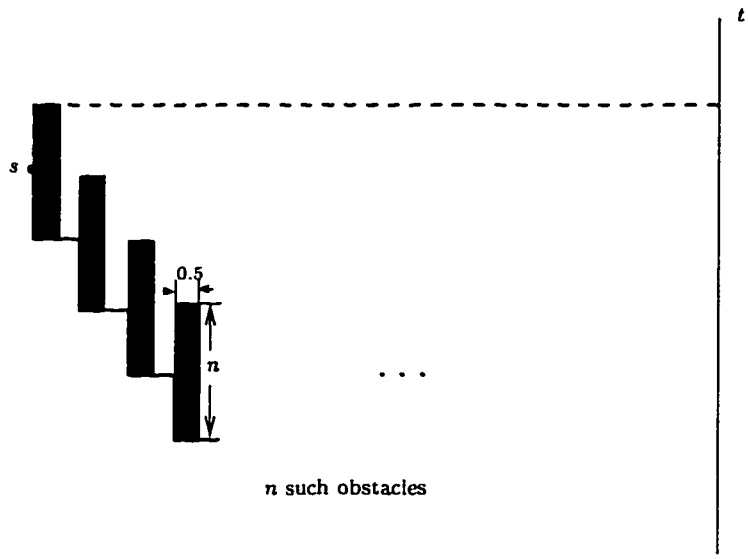

Figure 1.3: A scene where the greedy strategy has a competitive ratio of $\Omega(n)$.

problem, matching the lower bound of $\Omega(\sqrt{n})$ shown by Papadimitriou and Yanankakis [81]. Notice that a simple greedy strategy of going around each obstacle through the nearest corner does not work. In the scenario depicted in Figure 1.3, the greedy strategy is forced to have a ratio of $O(n)$, since the path taken by the robot (in solid thick line) has a length of $\Omega(n^2)$, while the optimal path (in dashed line) has a length of $O(n)$. A randomized algorithm given by Berman *et al.* [10] achieved a better ratio of $O(n^{4/9} \log n)$. However, there is still a big gap between this upper bound and the known lower bound of $\Omega(\log n)$ for randomized algorithms, given by Karloff *et al.* [54].

Another related navigation problem is called the *room problem* [11]. In this problem, the obstacles are oriented rectangles confined to lie within a square "room", and the target is a point in the room. An optimal algorithm was given by Bar-Eli *et al.* [6], whose competitive ratio is $\Theta(\log n)$, where n is the size of

the room. Chan and Lam [20] studied the point-to-point navigation problem with rectangle obstacles of a bounded aspect ratio.

There has also been work on other types of on-line navigation problems, for examples, the *searching problem*, where the goal is to locate instead of reaching some target [27, 58, 59], and the *mapping problem*, where the goal is to gain full information of an environment [29, 49, 50, 51].

In Chapter 4, we make extensions to the wall problem to deal with penetrable obstacles. In the *wall problem with penetrable obstacles*, each obstacle has a density (or weight) associated with it, and the objective is to minimize the weighted Euclidean distance. This problem models the situation where the robot travels in a field filled with lakes, swamps and hills. These can be considered as obstacles that are penetrable, but require more *effort per unit length* upon traveling through. We want to minimize the total effort spent to get to the target. We show that we can generalize the *sweeping strategy* in [11] to obtain an optimal on-line algorithm with a competitive ratio of $\Theta(\sqrt{n})$, where n is the Euclidean distance between the start point and the target. We further generalize this problem to the *recursive wall problem*, where within an obstacle, there may be obstacles with higher densities. An example situation modeled by this problem is when the robot climbs a mountain, it gets more energy consuming as it climbs higher.

Chapter 2

Kinodynamic Motion Planning

2.1 Introduction

This chapter contains two major results. The first one is a novel approximation method for the coupled kinodynamic motion-planning problem, and the second one is the first known polynomial-time approximation algorithm for the curvature-constrained shortest-path problem in three and higher dimensions. The second result is an application of the techniques developed for the kinodynamic problem. The discussion of the second result is contained in Section 2.6 while the rest of the sections focus on the kinodynamic motion-planning problem.

Let B be a point robot in a d-dimensional configuration space W. Without loss of generality, we can assume that W is a d-dimensional cube of size L. A *state* x of B is a pair $(\text{LOC}(x), \text{VEC}(x))$, where $\text{LOC}(x)$ is a point representing the location of B in the d-dimensional configuration space and $\text{VEC}(x)$ is a vector representing the velocity of B. The accelerations and velocities of B are both

bounded by 1 in L_2-norm. (If the bounds are any other numbers, we can scale them to 1; see Section 2.8) A trajectory (see Section 1.6.1 for its definition) is a $(\bar{a}, \bar{v})$-trajectory if at any time during the trajectory, the acceleration is bounded by $\bar{a}$ and the velocity is bounded by $\bar{v}$, both in L_2-norm. Let Ω be a set of configuration obstacles defined by a total of n constraints, where each constraint can be expressed by a polynomial of constant degree. A trajectory is *collision-free* if its path does not intersect the interior of Ω. A trajectory from a state x to a state y is *optimal*, if it is a collision-free $(1, 1)$-trajectory with a minimum time length, where the minimum is taken over all collision-free $(1, 1)$-trajectories from x to y. The kinodynamic motion-planning problem is to compute such optimal trajectories.

As we have seen that computing exact solutions is hard, we focus on developing fast approximation algorithms. To discuss about approximation solutions, the notion of a safe trajectory needs to be introduced. A point p has a *clearance* of μ if for any point $q \in \Omega$, $\|p - q\|_\infty \geq \mu$. A path is *$\mu$-safe* if for any point p along the path p has a clearance of μ. A trajectory is *μ-safe* if its associated path is μ-safe. A trajectory is an *optimal μ-safe* trajectory from x to y if its time length is the minimum over all μ-safe $(1, 1)$-trajectories from x to y. We will compute collision-free trajectories whose time length is within a factor of $(1 + \varepsilon)$ of the time length of optimal safe trajectories.

The first result of this chapter is summarized in the following theorem.

Theorem 2.1.1 *Fix an $l > 0$ and an $\varepsilon > 0$, after some pre-computation, we can achieve the following.*

Let W be a d-dimensional configuration space of size L. Let Ω be a set of configuration obstacles defined by a total of n algebraic equations, each of $O(1)$ degrees. Given any two states i and f, we can compute a collision-free $(1,1)$-trajectory from i' to f', such that the time length of the trajectory is at most $(1+\varepsilon)$ times the time length of an optimal $3l$-safe $(1,1)$-trajectory from i to f. Furthermore, $\|\text{LOC}(i') - \text{LOC}(i)\|_\infty \leq \varepsilon l$, $\|\text{VEC}(i') - \text{VEC}(i)\|_\infty \leq \varepsilon$, $\|\text{LOC}(f') - \text{LOC}(f)\|_\infty \leq \varepsilon l$ and $\|\text{VEC}(f') - \text{VEC}(f)\|_\infty \leq \varepsilon$.

The running time of our algorithm is $O(nN + N \log N(1/\varepsilon)^{4d-2})$, where $N = O((L/l)^d)$.

2.2 Preliminaries

2.2.1 Definitions and results from previous work

In this chapter, we generally use $\|\cdot\|_\infty$ to denote the L_∞-norm operation, and $\|\cdot\|$ the L_2-norm one.

Given a trajectory Γ, let $T(\Gamma)$ be the time length of Γ. Let $p_\Gamma(t), v_\Gamma(t), a_\Gamma(t)$, for $0 \leq t \leq T(\Gamma)$, be the location, velocity, and acceleration of Γ, respectively, at time t.

Let Π be a path and A a subset of the configuration space. We say that $d(\Pi, A) \leq \rho$, for some real number $\rho \geq 0$, if for every point $p \in \Pi$, there is a point $q \in A$ such that $\|p - q\|_\infty \leq \rho$, and $d(\Pi, A) = 0$ if $\Pi \in A$. Similarly, for two paths Π and Π', we say that $d(\Pi', \Pi) \leq \rho$, if for every point $p \in \Pi'$, there is a point $q \in \Pi$ such that $\|p - q\|_\infty \leq \rho$. Let Γ and Γ' be two trajectories. $d(\Gamma', \Gamma) \leq \rho$ (resp. $d(\Gamma', A) \leq \rho$), if $d(\Pi', \Pi) \leq \rho$ (resp. $d(\Pi', A) \leq \rho$), where

Π' and Π are the paths of Γ' and Γ respectively, and A is a subset of the configuration space.

For a state x, a state $x' \in ngb(x, \rho, \nu)$ if $\|\text{LOC}(x') - \text{LOC}(x)\|_\infty \leq \rho$ and $\|\text{VEC}(x') - \text{VEC}(x)\|_\infty \leq \nu$. Notice that if $x' \in ngb(x, \rho, \nu)$, then $x \in ngb(x', \rho, \nu)$.

Lemma 2.2.1 (Hollerbach [47]) *Let Π be the path of a $(\bar{a}, \bar{v})$-trajectory Γ. Π can be traversed by a $(\bar{a}/(1+\varepsilon)^2, \bar{v}/(1+\varepsilon))$-trajectory Γ' in time $(1+\varepsilon)\,T(\Gamma)$, for any $\varepsilon > 0$.*

The following corollary states a result of the TC-graph method in [32]; a brief description of the method can be found in Appendix A.

Corollary 2.2.2 (Donald and Xavier [32]) *Let W be a d-dimensional configuration space of size L. Let Γ be a $(1,1)$-trajectory from state i to state f such that Γ lies inside W. Given any $\varepsilon > 0$ and $\rho = O(1)$, applying the TC-graph method with appropriate parameters, we can compute in time $O(L^d(1/\varepsilon)^{6d-1})$ a trajectory Γ' from a state i' to a state f', such that $i' \in ngb(i, \varepsilon\rho/2, \varepsilon/2)$, $f' \in ngb(f, \varepsilon\rho/2, \varepsilon/2)$, $T(\Gamma') \leq (1+\varepsilon)\,T(\Gamma)$, and $d(\Gamma', W) \leq \varepsilon\rho/2$.*

In the rest of the paper, we fix $l > 0$ and $\epsilon > 0$ unless otherwise stated, where l gives the size of the smallest box in the box decomposition and ϵ describes the fineness of discretization.

2.2.2 Overview of the algorithm

Following the framework of many motion planning algorithms, our algorithm, termed as CS-graph (Configuration-State space-graph) method, reduces the

problem to that of computing and then searching on a graph. The nodes of the graph correspond to a set of non-uniform grid states (see Section 2.3). These grids are non-uniform in the sense that the discretization is coarser in regions that are farther away from any obstacles.

Section 2.5 describes how to compute the edge set of the graph. Each edge corresponds to a collision-free, near-optimal trajectory between two nodes. It is shown in this section that the graph satisfies the property that if there is an optimal safe trajectory, then there exists a path in the graph that corresponds to a collision-free trajectory whose time length is near optimal. Thus the problem is reduced to a shortest-path search on the computed graph.

In order to compute the edge set efficiently, we perform pre-computations. We derive a set of canonical trajectories (Section 2.4), which are sufficient for building the graph. These trajectories, once computed, can be used repeatedly for different problem instances with different obstacle arrangements.

A technical lemma, the Correcting Lemma, is described in Section 2.7. This lemma is a precise statement of our intuition that the discretization can be coarser in regions farther away from any obstacles.

2.3 Non-Uniform Grids

2.3.1 Box decomposition

Let $\tilde{\Xi}(s)$ be a set $\{p = (p_1, \ldots, p_d) \mid -s/2 \leq p_1, \ldots, p_d \leq s/2\}$. $\tilde{\Xi}(s)$ is called a d-dimensional *canonical box* of size s. Each set of $\{p \mid p \in \tilde{\Xi}(s), \text{ and } p_j = -s/2\}$ and $\{p \mid p \in \tilde{\Xi}(s), \text{ and } p_j = s/2\}$, for $1 \leq j \leq d$, is called a *face* (of size s) of

$\tilde{\Xi}(s)$. Ξ is a d-dimensional *box* of size s if it is a region which can be obtained from $\tilde{\Xi}(s)$ by translation, rotation and reflection. Similarly we can define the faces of Ξ. Notice that a face of a d-dimensional box Ξ of size s is really a $(d-1)$-dimensional box of size s. A d-dimensional box has $2d$ faces.

Given a d-dimensional box of size s, we can decompose it into 2^d boxes, each of size $s/2$. For each of them, we can further decompose it into 2^d boxes of size $s/4$ and so on. We stop further decomposing until certain conditions hold. We refer to this procedure, as well as the collection of undecomposed boxes obtained, as a *box decomposition.* When we refer to a box of a decomposition, we mean an undecomposed box.

Let W be a d-dimensional configuration space (assuming without loss of generality that W is a d-dimensional box) of size L, with a set Ω of configuration obstacles. (We consider the initial and final locations also as obstacle vertices.) A box is *free* if it does not intersect the interior of Ω. A box is *occupied* if it is a subset of Ω. Otherwise a box is called *partially occupied.* A box B_1 is *adjacent* to a box B_2 (or B_1 and B_2 are *neighbors*) if B_1 and B_2 share common points. We perform box decomposition on W until the following conditions are satisfied:

C1. The size of every partially occupied box is l.

C2. If a free box is adjacent to a non-free box, or to a free box with non-free boxes as neighbors, its size is l.

C3. The decomposition is *balanced*, i.e., a free box can have adjacent free boxes whose sizes are either twice as large or half as small.

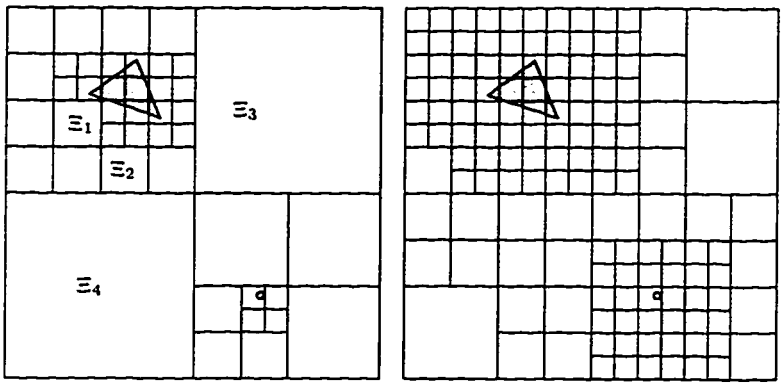

Figure 2.1: (a) Box decomposition after the first stage; (b) After the second stage.

Figure 2.1 shows an example of a box decomposition in 2D. The shaded triangle is an obstacle, and the shaded disk represents the initial location. Figure 2.1(a) gives a box decomposition only satisfying C1. In Figure 2.1(b), boxes are further decomposed to satisfy C2 (e.g. boxes Ξ_1 and Ξ_2) and C3 (e.g. boxes Ξ_3 and Ξ_4).

We use a tree structure to represent the box decomposition, where an internal node represent a decomposed box, and a leaf node an undecomposed one. Each internal node has 2^d children. We can perform the box decomposition in two stages. In the first stage, decompose until C1 is satisfied. Let M be the number of boxes obtained after this stage. It is easy to see that this stage can be computed in time $O(nM)$, where n is the number of obstacle constraints. In the second stage, we further decompose certain boxes until C2 and C3 are satisfied. Basically, we check all the leaf nodes in a bottom-up manner, working from small boxes to large ones. For each leaf node, find its neighbors and further decompose them if their sizes are too large. Let N be the size

of the final box decomposition. We claim that the time spent in this stage is $O(Nlog(L/l))$. Since the size of the smallest box is l, the box-decomposition tree has a depth of $O(\log(L/l))$. Thus finding the neighbors of a box takes at most $O(\log(L/l))$ time. Since we find neighbors for at most N boxes, the total running time is $O(N \log(L/l))$.

Notice that in the worst case, $N = O((L/l)^d)$, but N tends to be much smaller for cases where the obstacles are sparse.

Let $\mathcal{B}$ be the final box decomposition of W. Later, when we refer to a box (resp. a face) of $\mathcal{B}$, we mean an undecomposed box (resp. face) of $\mathcal{B}$.

2.3.2 Discretization

Let $\mathcal{B}$ be the box decomposition of W (with respect to Ω) satisfying C1–C3. For each face Δ of $\mathcal{B}$ such that Δ is not further decomposed (recall that Δ is a $(d-1)$-dimensional box), we select $(2/\epsilon)^{d-1}$ uniformly spaced points on Δ, with spacing $\epsilon s/2$, where s is the size of Δ. The set of points obtained in this way is called the *CS-grid points* of $\mathcal{B}$. Notice that the number of grid points on each face is the same, while the spacing of grid points grows linearly with the size of the faces. Thus we obtain a non-uniform discretization of the configuration space.

Let

$$\mathcal{V} = \{v = (j_1\epsilon, \ldots, j_d\epsilon) \mid j_1, \ldots, j_d \in \mathbb{Z} \text{ and } \|v\| \leq 1\}. \tag{2.1}$$

$\mathcal{V}$ is called the set of *grid velocities*. A state a is called a *CS-grid state* of $\mathcal{B}$ if $\text{LOC}(a)$ is a CS-grid point of $\mathcal{B}$ and $\text{VEC}(a) \in \mathcal{V}$. The grid states of $\mathcal{B}$ (including the initial and final state) are the nodes of the graph to be constructed.

For a state a that lies on a face Δ, let $s(a)$ be the function returning the size of the face Δ. Define $ngb(a) = ngb(a, \epsilon s(a)/4, \epsilon/2)$. By the construction of the grid states, it is easy to show

Lemma 2.3.1 *If a is a state such that* LOC(a) *lies on a face Δ of $\mathcal{B}$, there exists a CS-grid state a' such that $a \in ngb(a')$.*

2.4 Precomputing Canonical Trajectories

2.4.1 Extended boxes

Let $\mathcal{B}$ be the box decomposition of W. Given a state a, define $\xi(a)$ to be the *extended box* of a, which is the smallest region comprised of boxes such that i) The boundary of the union of these boxes form a closed $(d-1)$-dimensional surface; and ii) For any point q that lies on the closed surface, $\|\text{LOC}(a) - q\|_\infty \geq s(a)/2$.

Let Δ be a face of $\mathcal{B}$, of size s, and let Ξ be the box that contains Δ (pick one arbitrarily if both of the two boxes containing Δ have size s). Decompose Δ into 2^{d-1} $(d-1)$-dimensional boxes, each of size $s/2$, and label them $\Delta_1, \ldots, \Delta_{2^{d-1}}$. Given any two states a and b, if both LOC(a) and LOC(b) lie on Δ_j, then $\xi(a)$ and $\xi(b)$ are the same. This implies that we can extend the definition of extended boxes for faces. Define $\xi(\Delta_j)$ to be the extended boxes of Δ_j, which is the same as the extended boxes of $\xi(a)$, for any state a such that LOC(a) lies in Δ_j. $\xi(\Delta_j)$ is called an extended box of *size* $s/2$, since Δ_j is a face of size $s/2$.

An extended box (resp. a box) is said to have a *clearance* of μ, if for any

point p that lies in the extended box (resp. the box), p has a clearance of μ. Notice that a free box has a clearance of s, if all its neighbors are free boxes and have a size of at least s.

Lemma 2.4.1 *An extended box of size $s \geq 2l$ has a clearance of at least $s/2$; an extended box of size l has a clearance of at least l.*

Proof: Suppose that the extended box is an extended box of face Δ_j whose size is s. Let Ξ be the box containing Δ_j, and the size of Ξ is $2s$. Each other box in the extended box is a neighbor of Ξ. Thus their sizes are at least s, by the balanced property of the box decomposition.

If $s \geq 2l$, for any box Ξ' in the extended box, its neighbors must have sizes $\geq s/2$. By property C2 of the box decomposition, these neighbor boxes must be free. Otherwise, the size of Ξ' can not be larger than l. Thus Ξ' has a clearance of $\geq s/2$. This implies that the extended box has a clearance of $\geq s/2$.

If $s = l$, Ξ is a box of size $2l$. Since the box decomposition satisfies C2, we know that all Ξ's neighbors and all their neighbors are free boxes whose sizes are at least l. This implies that the extended box has a clearance of l. This completes the proof. □

Lemma 2.4.2 *Let $\xi(\Delta_j)$ be an extended box of a face Δ_j of size $l/2$. If there exists a point p on Δ_j such that p has a clearance of $3l$, then the clearance of the extended box is at least l.*

Proof: It suffices to prove that for each point q on the boundary of $\xi(\Delta_j)$, q has a clearance of l. If q lies on the face of a box of size l, then $\|q - p\|_\infty \leq 2l$.

Since the clearance of p is $3l$, the clearance of q is at least $3l - 2l = l$. If q lies on the face of a box of size $\geq 2l$, its clearance is at least l, since the clearance of this box is at least l. □

Let $\tilde{\Delta}(s/2) = \{p = (p_1, \ldots, p_d) \mid p_1 = -s/2, \text{and } 0 \leq p_2, \ldots, p_d \leq s/2\}$. $\tilde{\Delta}(s/2)$ is called the *canonical face* of size $s/2$. We can transform Δ_j, Ξ and $\xi(\Delta_j)$, by translation, rotation and reflection, such that Ξ becomes $\tilde{\Xi}(s)$, the canonical box of size s, and Δ_j becomes $\tilde{\Delta}(s/2)$. The transformed extended box $\xi(\Delta_j)$ is called a *canonical extended box* of $\tilde{\Delta}(s/2)$, or a canonical extended box of size $s/2$. Since the box decomposition $\mathcal{B}$ is balanced, the number of canonical extended boxes of a fixed size is a function, depending only on the number of dimensions d, but not on the size, or the location. Let $\xi(s)$ be the set of canonical extended boxes of $\tilde{\Delta}(s)$. We can give an order to $\xi(s)$ and let $\xi(s, j)$ be the jth canonical extended box of $\tilde{\Delta}(s)$, for some valid j. Figure 2.2 gives some examples of canonical extended boxes in 2D.

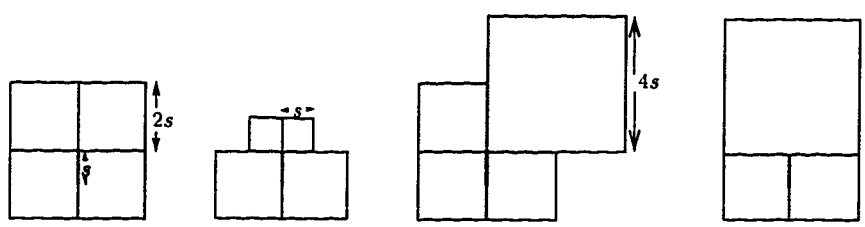

Figure 2.2: Some canonical extended boxes of $\tilde{\Delta}(s)$, where $\tilde{\Delta}(s)$ is a line segment drawn in thick lines.

Let $\mathcal{P}(s)$ be the set of $(1/\epsilon)^{d-1}$ uniformly spaced points on $\tilde{\Delta}(s)$, with spacing ϵs. Define $\mathcal{Q}(s, j)$ as follows. If Δ is a face of size s' belonging to the boundary of $\xi(s, j)$, include the set of $(2/\epsilon)^{d-1}$ uniformly spaced points with

spacing $\epsilon s'/2$ on Δ in $\mathcal{Q}(s,j)$. Let $\mathcal{I}(s) = \mathcal{P}(s) \times \mathcal{V}$, and $\mathcal{F}(s,j) = \mathcal{Q}(s,j) \times \mathcal{V}$. A state in the set $\mathcal{I}(s)$ is called a *canonical starting state*, and a state in the set $\mathcal{F}(s,j)$ is called a *canonical ending state* of $\xi(s,j)$. It is easy to see that $|\mathcal{I}(s)| = O((1/\epsilon)^{2d-1})$. It also holds that $|\mathcal{F}(s,j)| = O(d(1/\epsilon)^{2d-1})$. This is because of the balanced property of the box decomposition. The boundary of $\xi(s,j)$ can contain at most $O(d)$ faces, each contributing $O((1/\epsilon)^{2d-1})$ states.

2.4.2 Computing connection tables

Let $\xi(s,j)$ be the jth canonical extended box of $\tilde{\Delta}(s)$, for some valid s and j. Let $CT(s,j)$ be the *connection table* of $\xi(s,j)$. The connection table contains pre-computed trajectories which connect canonical starting states and canonical ending states. In this section we describe how to compute the connection tables.

First we introduce our Correcting Lemma (whose proof can be found in Section 2.7), which states a result essential to the proofs of other lemmas.

Lemma 2.4.3 (Correcting Lemma) *Fix a constant $c \leq 1$, and let $\rho_c = \bar{v}^2/(c\bar{a})$, where $\bar{a}, \bar{v} > 0$ are arbitrary. Let Γ be an $(\bar{a}, \bar{v})$-trajectory from i to f. Given any $\rho > 0$ and $\varepsilon > 0$, and given any two states g and h, if $\rho > \rho_c$, $\|\text{LOC}(f) - \text{LOC}(i)\|_\infty \geq \rho$, $g \in ngb(i, \varepsilon\rho/2, \varepsilon\bar{v}/2)$, and $h \in ngb(f, \varepsilon\rho/2, \varepsilon\bar{v}/2)$, we can construct a trajectory Γ' from g to h, by correcting Γ, such that Γ' satisfies the following properties:*

P1. $T(\Gamma') = T(\Gamma)$.

P2. Γ' *is a* $((1 + 4c\sqrt{d}\varepsilon)^2\bar{a}, (1 + 4c\sqrt{d}\varepsilon)\bar{v})$*-trajectory.*

P3. $d(\Gamma', \Gamma) \leq (17/16)\varepsilon\rho$.

Here onwards, we fix

$$c = \max(2/l, 1), \tag{2.2}$$

where l is the size of the smallest box. In our later application of this lemma, it always holds that $\bar{v}^2/\bar{a} = 1$, thus $\rho_c = 1/c$. This implies that

$$l/2 \geq \rho_c. \tag{2.3}$$

Also let

$$e = 4c\sqrt{d}. \tag{2.4}$$

Consider a pair of states (i, f) that lie on the boundary of an extended box ζ. A $(1, 1)$-trajectory from i to f is called *legal* if its path lies inside ζ. The pair (i, f) is *legal* if there exists at least a legal trajectory from i to f. We will see later that these legal trajectories are the potential trajectories that we need to approximate. We call a pair of grid state (i, f) *good* if there exists at least a legal pair (g, h) such that $g \in ngb(i)$ and $h \in ngb(f)$. Roughly speaking, the next lemma (Existing Lemma) states that if (i, f) is a good pair, then there *exists* a trajectory Γ from i to f such that Γ approximates *any* legal trajectories for any legal pair (g, h), where $g \in ngb(i)$ and $h \in ngb(f)$. Thus for our purpose of approximation, it is enough to compute only the trajectories for good pairs. For a pair of legal states (a, b), let $\hat{T}(a, b)$ be the time length of the legal trajectory from a to b whose time length is the smallest among all legal trajectories from a to b.

Lemma 2.4.4 (Existing Lemma) *Let $\xi(s, j)$ be the jth canonical extended box of $\tilde{\Delta}(s)$, for some valid s and j. For any $i \in \mathcal{I}(s)$ and $f \in \mathcal{F}(s, j)$, if (i, f)*

is good, then there exists a trajectory Γ *from* i *to* f*, such that the trajectory satisfies the following properties:*

P1. For any $i' \in ngb(i)$*, and* $f' \in ngb(f)$*, if* (i', f') *is legal,* $T(\Gamma) \leq \hat{T}(i', f')$.

P2. Γ *is a* $((1+e\epsilon)^2, (1+e\epsilon))$*-trajectory.*

P3. $d(\Gamma, \xi(s,j)) \leq (17/8)\epsilon s$.

Proof: Let $g \in ngb(i)$ and $h \in ngb(f)$ be such that (g,h) is legal and that $\hat{T}(g,h)$ is the smallest among those of all such legal pairs; let Γ' be the $(1,1)$-trajectory from g to h whose time length is $\hat{T}(g,h)$. We will show the existence of a trajectory satisfying P1–P3, by constructing one from Γ'. There are three cases depending on whether $s(f)$ (the size of the face where f lies) is s, $2s$ or $4s$. We will prove for the case when $s(f) = 4s$; this case gives the worst bounds. The other two cases can be handled in a similar way.

Let Γ be the trajectory obtained by correcting Γ'. Let $\bar{a} = 1$, $\bar{v} = 1$, and $\rho = 2s$ in the Correcting Lemma. Since $s \geq l/2$, $\rho = 2s \geq l > \rho_c$. We can show that $\|\text{LOC}(h) - \text{LOC}(g)\|_\infty \geq 2s = \rho$. Since $g \in ngb(i)$, $g \in ngb(i, \epsilon\, s(i)/4, \epsilon/2)$. Since $\rho = 2s = s(i)$, it is also true that $g \in ngb(i, \epsilon\rho/2, \epsilon/2)$. This implies that $i \in ngb(g, \epsilon\rho/2, \epsilon/2)$. Similarly, $f \in ngb(h, \epsilon\rho/2, \epsilon/2)$. Thus the conditions of the Correcting Lemma are satisfied. This implies that $T(\Gamma) \leq \hat{T}(g,h)$, satisfying P1. Γ is a $((1+e\epsilon)^2, (1+e\epsilon))$-trajectory, and $d(\Gamma, \Gamma') \leq (17/16)\epsilon\rho = (17/8)\epsilon s$. Since $d(\Gamma', \xi(s,j)) = 0$, $d(\Gamma, \xi(s,j)) \leq (17/8)\epsilon s$, proving P3. □

For a good pair (i,f) (with respect to $\xi(s,j)$), our goal is to approximate (since we do not know how to compute exactly) such a trajectory Γ as stated in

the Existing Lemma. The approximation is done in two steps. First, we apply the TC-graph method to compute a trajectory Γ' which is close to Γ time wise and spatial wise. However, the TC-graph method does not guarantee that Γ' is from i to f. Instead, we only know that Γ' is from some i' close to i to some f' close to f. Second, we correct Γ' to obtain a trajectory Γ'' which is really from i to f. By the Correcting Lemma, Γ'' approximates Γ', and thus approximates Γ. In the first step, by setting $\rho = l/2$ and $\varepsilon = \epsilon$ in Corollary 2.2.2, we are able to guarantee that

- Γ' is a $((1+e\epsilon)^2, (1+e\epsilon))$-trajectory. (Notice that even though Corollary 2.2.2 is about $(1,1)$-trajectories, the results apply to $((1+e\epsilon)^2, (1+e\epsilon))$-trajectories.)
- $T(\Gamma') \leq (1+\epsilon)T(\Gamma)$.
- Γ' is from some $i' \in ngb(i, \epsilon l/4, \epsilon/2)$ to some $f' \in ngb(f, \epsilon l/4, \epsilon/2)$.
- $d(\Gamma', \xi(s,j)) \leq (17/8)\epsilon s + \epsilon l/2$. The existence of a trajectory satisfying this condition in the TC-graph is due to the fact that $d(\Gamma, \xi(s,j)) \leq (17/8)\epsilon s$, and the existence of a trajectory $\tilde{\Gamma}$ in the graph such that $d(\tilde{\Gamma}, \Gamma) \leq \epsilon l/2$.

In the second step, setting $\bar{a} = (1+e\epsilon)^2$, $\bar{v} = 1+e\epsilon$, and $\rho = l/2 \geq \rho_c$, we can see that the conditions of the Correcting Lemma are satisfied. Thus Γ'' is a $((1+e\epsilon)^4, (1+e\epsilon)^2)$-trajectory, $T(\Gamma'') = T(\Gamma') \leq (1+\epsilon)T(\Gamma)$ and $d(\Gamma'', \Gamma') \leq (17/32)\epsilon l$. Since $d(\Gamma', \xi(s,j)) \leq (17/8)\epsilon s + \epsilon l/2$, $d(\Gamma'', \xi(s,j)) \leq (17/8)\epsilon s + (33/32)\epsilon l \leq 5\epsilon s$ (using the fact that $s \geq l/2$). In summary, we

obtain

Lemma 2.4.5 (Loose Tracking Lemma) *Let $\xi(s,j)$ be the jth canonical extended box of $\tilde{\Delta}(s)$, for some valid s and j. Let $i \in \mathcal{I}(s)$ and $f \in \mathcal{F}(s,j)$. If (i,f) is good, then Γ'' computed as above satisfies the following properties:*

P1. For any $g \in ngb(i)$, and $h \in ngb(f)$, if (g,h) is legal, $T(\Gamma'') \leq (1+\epsilon)\hat{T}(g,h)$.

P2. Γ'' is a $((1+e\epsilon)^4,(1+e\epsilon)^2)$-trajectory.

P3. $d(\Gamma'',\xi(s,j)) \leq 5\epsilon s$.

Fix a canonical extended box $\xi(s,j)$, for some valid s and j. For each $i \in \mathcal{I}(s)$ and $f \in \mathcal{F}(s,j)$, we pre-compute a trajectory from i to f as described above. We store the trajectory (i.e., the initial state i, the final state f, the acceleration function, and the time length) in the connection table $CT(s,j)$. A connection table computed in this way satisfies the following property:

Lemma 2.4.6 (Connection Table Property) *Let $\xi(s,j)$ be the jth canonical extended box of $\tilde{\Delta}(s)$, for some valid s and j, and let $CT(s,j)$ be the connection table for $\xi(s,j)$. For any canonical starting state $i \in \mathcal{I}(s)$ and any canonical ending state $f \in \mathcal{F}(s,j)$, if (i,f) is good, then $CT(s,j)$ contains a trajectory Γ from i to f and Γ satisfies the following properties:*

P1. For any $g \in ngb(i)$, and $h \in ngb(f)$, if (g,h) is good, $T(\Gamma) \leq (1+\epsilon)\hat{T}(g,h)$.

P2. Γ is a $((1+e\epsilon)^4,(1+e\epsilon)^2)$-trajectory.

P3. $d(\Gamma, \xi(s,j)) \leq 5\epsilon s$.

Let $\xi(l/2,0)$ be the canonical extended box that consists of 4 boxes of size l. Apart from $CT(l/2,0)$, we maintain a special connection table CT_0 for this canonical extended box. For each $i \in \mathcal{I}(l/2)$, we construct a TC-graph G rooted at i (for a description of the TC-graph method, see A). We add a trajectory Γ from i to f in CT_0, if 1) f is a node of G, 2) f either lies inside $\xi(l/2,0)$, or its L_∞ distance to the extended box is no more than $\epsilon l/2$, 3) there is a path on G from i to f (whose corresponding trajectory is Γ). By the property of the TC-graph method, we can show the following

Lemma 2.4.7 *Let $i \in \mathcal{I}(l/2)$. For any pair of states g and h, such that $g \in ngb(i, \epsilon l/2, \epsilon/2)$, that h lies inside $\xi(l/2,0)$, and that the optimal $(1,1)$-trajectory from g to h lies inside $\xi(l/2,0)$, there exists a $(1,1)$-trajectory in CT_0 from i to some f, such that $f \in ngb(h, \epsilon l/2, \epsilon/2)$, and that the time length of this trajectory is no more than $(1+\epsilon)$ times the time length of the optimal $(1,1)$-trajectory from g to h.*

To correct a trajectory takes only $O(1)$ time, since it basically involves computing and adding $O(1)$ number of corrective acceleration terms (see Section 2.7). So the time complexity of computing a connection table $CT(s,j)$ is determined by how fast we can compute a trajectory Γ' for each pair of canonical starting and ending states. We can compute the trajectories using the TC-graph method in the following manner. For each canonical starting state i, compute the TC graph rooted at i. Expand the graph until all the canonical ending states of $\xi(s,j)$ are reached, or no new nodes can be added.

A shortest graph path from i to a canonical ending state gives a trajectory between these two states. All these paths can be stored in a concise way by storing at each node its preceding node along the paths. Since the extended box is of size s, the graph has $O(s^d(1/\epsilon)^{6d-1})$ edges. This also bounds the time and space for computing the trajectories for a single canonical starting state. Thus it takes $O(s^d(1/\epsilon)^{8d-2})$ time and space to compute a connection table $CT(s,j)$. A Similar analysis shows that CT_0 can also be computed in $O(l^d(1/\epsilon)^{8d-2})$ time and space.

2.5 Approximation in Presence of Obstacles

2.5.1 The algorithm

In this section we present our approximation algorithm for computing collision-free near-optimal trajectories in presence of obstacles. Let W be a d-dimensional configuration space of size L, with a set Ω of obstacles. Let i and f be the initial and final states respectively. And let $\mathcal{B}$ be the box decomposition of W satisfying C1–C3. Our approximation algorithm constructs a weighted directed graph $G = (V, E)$, where V includes i, f and a subset of CS-grid states induced by $\mathcal{B}$. A CS-grid state a is included in V if i) $s(a) \geq 2l$; or ii) $s(a) = l$ and the clearance of $\xi(a)$ is at least l.

Let a and b be two states that lie within an extended box ζ. To see if there is a pre-computed trajectory from a to b, we transform ζ to its canonical form, and also a and b with it. If $\mathcal{T}$ is the transformation, we say that there is a pre-computed trajectory from a to b if there is a trajectory from $\mathcal{T} \circ a$ to $\mathcal{T} \circ b$

in the connection table of $\mathcal{T} \circ \zeta$. If Γ is the trajectory from $\mathcal{T} \circ a$ to $\mathcal{T} \circ b$, then $\mathcal{T}^{-1} \circ \Gamma$ is the trajectory from a to b. It is easy to see that $\mathcal{T}^{-1} \circ \Gamma$ satisfies the connection table properties P1–P3, with respect to ζ.

We construct the edge set in the following way. For node i, add an edge from i to another node a, if i) i lies within $\xi(a)$; and ii) there is a pre-computed trajectory from a^- to i^-.[1] Similarly, for node f, add an edge from a node a to f, if i) f lies within $\xi(a)$; and ii) there is a pre-computed trajectory from a to f. For any other node a, add an edge from a to a another node b, if i) b lies on the boundary of $\xi(a)$; and ii) there is a pre-computed trajectory from a to b. The weight of each edge is equal to the time length of its corresponding trajectory.

Theorem 2.5.1 (Safe Loose Tracking Theorem) *If there exists an optimal $3l$-safe $(1,1)$-trajectory Γ from state i to state f, then the graph G, constructed as above, contains a path whose corresponding trajectory Γ' satisfies the following properties:*

P1. $T(\Gamma') \leq (1+\epsilon)\, T(\Gamma)$.

P2. Γ' *is a* $((1+e\epsilon)^4, (1+e\epsilon)^2)$*-trajectory.*

P3. Γ' *does not intersect the interior of* Ω.

P4. Γ' *is from some* $i' \in ngb(i, \epsilon l/2, \epsilon/2)$ *to some* $f' \in ngb(f, \epsilon l/2, \epsilon/2)$.

Proof: Let Γ be divided into segments of trajectories, $\Gamma_{a_0a_1} \| \Gamma_{a_1a_2} \| \ldots \| \Gamma_{a_{m-1}a_m}$, such that $a_0 = i$, $a_m = f$, and each a_j is the state where Γ first exits $\xi(a_{j-1})$, for

[1] For a state a, we use a^- to denote a state of $(\text{LOC}(a), -\text{VEC}(a))$.

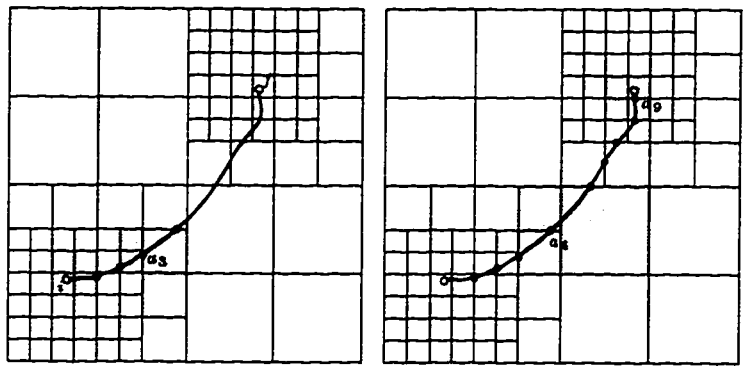

Figure 2.3: (a) $\xi(a_0)$ and $\xi(a_3)$; (b) $\xi(a_4)$ and $\xi(a_9)$.

$1 \leq j \leq m-1$. Figure 2.3 gives an example in 2D. The optimal $(1,1)$-trajectory from i to f (drawn in thick curve) is divided into 10 segments (each dark circle represents an a_j, for $1 \leq j < 10$). Some of the extended boxes are shown as shaded regions. We consider the following parts, $\Gamma_{a_0a_1}$, $\Gamma_{a_1a_2}\| \ldots \|\Gamma_{a_{m-2}a_{m-1}}$ and $\Gamma_{a_{m-1}a_m}$ separately.

Since each a_j, for $1 \leq j \leq m-1$, is on some face of $\mathcal{B}$, by Lemma 2.3.1, we can find CS-grid state b_j, such that $a_j \in ngb(b_j)$. If $s(a_j) \geq 2l$, then $s(b_j) \geq 2l$ and b_j is in the node set V. Otherwise, since a_j is $3l$-safe, $\xi(a_j)$ has a clearance of at least l. Since $\xi(b_j) = \xi(a_j)$, $\xi(b_j)$ also has a clearance of at least l and b_j is in the node set V. We will show that the graph path $(b_1, b_2)\| \ldots \|(b_{m-2}, b_{m-1})$ corresponds to a trajectory satisfying P1–P3. To this end, we show that there is an edge from b_{j-1} to b_j whose corresponding trajectory $\Gamma_{b_{j-1}b_j}$ satisfies P2, P3, and $T(\Gamma_{b_{j-1}b_j}) \leq (1+\epsilon)\,T(\Gamma_{a_{j-1}a_j})$, for $1 < j < m$.

By the way Γ is divided, $d(\Gamma_{a_{j-1}a_j}, \xi(a_{j-1})) = 0$. Thus each pair (a_{j-1}, a_j) is legal. This implies that each (b_{j-1}, b_j) is good. By the Connection Table

Property, there is a pre-computed trajectory $\Gamma_{b_{j-1}b_j}$ from b_{j-1} to b_j. Thus an edge from b_{j-1} to b_j is added in the construction of G. Also by the Connection Table Property, $\Gamma_{b_{j-1}b_j}$ is a $((1+e\epsilon)^4, (1+e\epsilon)^2)$-trajectory, and its time length is no more than $(1+\epsilon)T(\Gamma_{a_{j-1}a_j})$. Next we need to show that $\Gamma_{b_{j-1}b_j}$ is collision-free.

Let s be the size of $\xi(b_{j-1})$. We consider the three cases $s \geq 2l$, $s = l$ and $s = l/2$ separately. If $s \geq 2l$ (resp. $s = l$), the extended box $\xi(b_{j-1})$ has a clearance of $s/2$ (resp. l). On the other hand, $d(\Gamma_{b_{j-1}b_j}, \xi(b_{j-1})) \leq 5\epsilon s$. Thus if $\epsilon \leq 1/10$ is chosen small enough, $\Gamma_{b_{j-1}b_j}$ is collision-free. When $s = l/2$, $\xi(b_{j-1})$ has a clearance of at least l. This is because a_{j-1} is $3l$-safe. By Lemma 2.4.1, $\xi(a_{j-1})$ has a clearance of at least l. This implies that $\xi(b_{j-1})$ has a clearance of at least l, since $\xi(b_{j-1}) = \xi(a_{j-1})$. Since $d(\Gamma_{b_{j-1}b_j}, \xi(b_{j-1})) \leq (5/2)\epsilon l$, $\Gamma_{b_{j-1}b_j}$ is collision-free if ϵ is chosen small enough.

Now consider the part $\Gamma_{a_{m-1}a_m}$, which lies inside $\xi(a_{m-1})$. When constructing $\mathcal{B}$, $\textsc{loc}(f)$ is considered as an obstacle vertex. Thus the box containing $\textsc{loc}(f)$ and its neighbors all have size l. This means that $\xi(a_{m-1})$ is an extended box consisting of 4 boxes of size l. By Lemma 2.4.7, there exists in CT_0 a pre-computed trajectory $\Gamma_{b_{m-1}f}$ from b_{m-1} to some $f' \in ngb(f, \epsilon l/2, \epsilon/2)$. Thus an edge from b_{m-1} to f is added in the construction of G. Also $\Gamma_{b_{m-1}f}$ is a $(1,1)$-trajectory whose time length is no more than $(1+\epsilon)\Gamma_{a_{m-1}a_m}$. To show that $\Gamma_{b_{m-1}f}$ is collision-free, notice that the clearance of $\xi(a_{m-1})$ is at least l, since a_{m-1} is $3l$-safe. But $d(\Gamma_{b_{m-1}f}, \xi(b_{m-1})) \leq \epsilon l/2$, this trajectory is also collision-free. We can show similar results for $\Gamma_{a_0a_1}$. This completes the proof of this theorem. □

Thus the approximation problem is transformed to one of searching for a shortest path on the graph G. Notice that in order for this algorithm to be correct, each edge introduced in G must correspond to a collision-free trajectory. This is guaranteed by the way we choose the node set V, and the connection table properties (a proof similar to the one showing that each $\Gamma_{b_{j-1}b_j}$ is collision-free, used in the above theorem). We do not have to explicitly check whether each trajectory segment is collision-free. The obtained shortest path corresponds to a $((1+e\epsilon)^4, (1+e\epsilon)^2)$-trajectory. Applying the Time-rescaling Lemma with a scaling factor of $(1+e\epsilon)^2$, we can obtain the following

Corollary 2.5.2 *Let W be a d-dimensional configuration space of size L, and Ω a set of obstacles. Let Γ be an optimal $3l$-safe $(1,1)$-trajectory from i to f. We can compute a $(1,1)$-trajectory Γ' from some $i' \in ngb(i, \epsilon l/2, 3e\epsilon)$ to some $f' \in ngb(f, \epsilon l/2, 3e\epsilon)$ such that $T(\Gamma')$ is at most $(1+3e\epsilon)$ times $T(\Gamma)$.*

2.5.2 Time complexity

The running time of the algorithm consists of the following components.

1. Time to generate the graph nodes (i.e., the grid states). If N is total number of boxes in the final decomposition, the time to perform the decomposition is $O(nN + N \log N)$. Since each box contributes at most $O((1/\epsilon)^{2d-1})$ grid states, the time to generate the grid states, after a decomposition, is $O(N(1/\epsilon)^{2d-1})$.

2. Time to compute the graph edges. Since each node is connected to at most $O((1/\epsilon)^{2d-1})$ other nodes, the total number of edges is $O(N(1/\epsilon)^{4d-2})$.

This bounds the time to compute the edges, since it takes $O(1)$ time to compute an edge.

3. Time to search for a shortest graph path. Using Dijkstra algorithm with the priority queue implemented with a binary heap, this time is $O((|V| + |E|) \log |V|)$, where $|V|$ are $|E|$ are the number of nodes and edges respectively. Plugging in our numbers, the time for searching is roughly $O(dN \log N(1/\epsilon)^{4d-2})$.

4. Time to rescale the obtained trajectory to a $(1, 1)$-trajectory, which is $O(1)$.

Combining Corollary 2.5.2 and the time-complexity analysis, and choosing small enough $\epsilon = O(\varepsilon)$, we obtain Theorem 2.1.1.

2.6 An Application to Curvature-Constrained Shortest-Path Problem

Given a set Ω of obstacles in a d-dimensional space, an initial state I and a final state F (where a state is a defined by a location and an orientation), the curvature-constrained shortest-path problem is to find a shortest path from I to F such that the path obeys the curvature constraint and does not intersect Ω. It has been observed in [39] that the curvature-constrained shortest-path problem is a restricted case of the kinodynamic motion planning problem, with the L_2 norms of the velocities fixed to be 1. However, if we require that the velocities of the approximation trajectories be fixed in L_2 norm, the techniques

developed so far for the general kinodyanmic case can not be applied to this restricted case. As pointed out in [30], a necessary condition for the techniques to apply is that the set of *feasible instantaneous accelerations* (accelerations that can be applied without violating the dynamic constraints) spans d dimensions. On the other hand, if the velocities are fixed in L_2 norm, the set of instantaneous accelerations spans only $d-1$ dimensions, because they have to be perpendicular to the instantaneous velocity.

Our contribution is that we look at the curvature-constrained shortest-path problem from a different view, which enables us to overcome the difficulty mentioned above. Basically, instead of requiring that the L_2 norms of the velocities be fixed to be 1, we only force them to fall within a small range close to 1. In this way, we are able to obtain a path whose maximum curvature is slightly larger than but can be arbitrarily close to 1.

In this section, we use lower-case letters to denote states as defined for the kinodynamic case, but upper-case letters (usually X, Y, U, V) to denote states whose velocities are 1 in L_2 norm (i.e., $\|\text{VEC}(X)\| = \|\text{VEC}(Y)\| = \|\text{VEC}(U)\| = \|\text{VEC}(V)\| = 1$). A path is called a *c-constrained* path if its average curvature is at most c. We say that Γ is a $(1, \tilde{1})$-trajectory if $\|a_\Gamma(t)\| \leq 1$ and $\|v_\Gamma(t)\| = 1$ for $0 \leq t \leq T(\Gamma)$. Given a path Π, let $L(\Pi)$ be its path length.

2.6.1 Approximating in absence of obstacles

The following lemma states a result similar to Corollary 2.2.2, but for the curvature-constrained case. This lemma enables us to apply the method developed in the previous sections to the curvature-constrained shortest-path

problem.

Lemma 2.6.1 *Let W be a d-dimensional configuration space of size L. Let Π be a 1-constrained path from state X to state Y and lying inside W. Given any $\varepsilon > 0$ and $\rho = O(1)$, we can compute in time $O(L^d(1/\varepsilon)^{6d-1})$ a path Π', such that Π' satisfies the following properties:*

P1. Π' *is* $(1+\varepsilon)$*-constrained.*

P2. $L(\Pi') \leq (1+\varepsilon)L(\Pi)$.

P3. $d(\Pi', W) \leq \varepsilon\rho/2$.

P4. Π' *is from some state* $X' \in ngb(X, \varepsilon\rho/2, \varepsilon/2)$ *to some state* $Y' \in ngb(Y, \varepsilon\rho/2, \varepsilon/2)$.

Proof: Let $\delta = \varepsilon/8$, $\eta_x = \varepsilon\rho/2$, and $\eta_v = \varepsilon/(8\sqrt{d})$.

We consider Π to be the path of a trajectory Γ, such that Γ is from $i = (\text{LOC}(X), \text{VEC}(X)/(1+\delta))$ to $f = (\text{LOC}(Y), \text{VEC}(Y)/(1+\delta))$, and that $\|v_\Gamma(t)\| = 1/(1+\delta)$. Notice that $T(\Gamma) = (1+\delta)\, L(\Pi)$. Since Π is a 1-constrained path, and at any time t, the curvature of Π is given by $\|a_\Gamma(t)\|/\|v_\Gamma(t)\|^2$, the acceleration of Γ is bounded by

$$\|a_\Gamma(t)\| \leq 1 \cdot \|v_\Gamma(t)\|^2 \leq 1/(1+\delta)^2.$$

This implies that Γ is a $(1/(1+\delta)^2, 1/(1+\delta))$-trajectory. Let $\mathcal{A}_\delta$ be the set of accelerations whose L_2 norms are bounded by $1/(1+\delta)^2$. This is the set of accelerations used by Γ.

Let $\mu = \kappa_l = \delta/(4\sqrt{d})$, and $\mathcal{A}_\mu$ the set of accelerations as defined in Appendix A (see Equation A.5 and A.6). Thus $\mathcal{A}_\mu$ has a uniform advantage of κ_l over $\mathcal{A}_\delta$. By the Tracking Lemma (see Appendix A), there exists a $\tau = O(\varepsilon)$ (satisfying Equation A.4), and a τ-bang trajectory Γ' using $\mathcal{A}_\mu$, such that $T(\Gamma') = T(\Gamma)$, and that Γ' tracks Γ to a tolerance of (η_x, η_v). This implies that

$$\|v_{\Gamma'}(t) - v_\Gamma(t)\|_\infty \leq \eta_v$$

$$\|v_{\Gamma'}(t) - v_\Gamma(t)\| \leq \sqrt{d}\eta_v$$

$$\|v_\Gamma(t)\| - \sqrt{d}\eta_v \leq \|v_{\Gamma'}(t)\| \leq \|v_\Gamma(t)\| + \sqrt{d}\eta_v$$

$$\frac{1}{1+\delta} - \sqrt{d}\frac{\varepsilon}{8\sqrt{d}} \leq \|v_{\Gamma'}(t)\| \leq \frac{1}{1+\delta} + \sqrt{d}\frac{\varepsilon}{8\sqrt{d}}$$

$$\frac{1}{1+\delta} - \frac{\varepsilon}{8} \leq \|v_{\Gamma'}(t)\| \leq \frac{1}{1+\delta} + \frac{\varepsilon}{8}$$

$$1 - \frac{\varepsilon}{4} \leq \|v_{\Gamma'}(t)\| \leq 1 + \frac{\varepsilon}{8}$$

Let Π' be the path of trajectory Γ'. We will show that Π' satisfies P1–P4. Since $\|a_{\Gamma'}(t)\| \leq 1$, this bounds the maximum curvature of Π' to be at most

$$\frac{1}{(1-\varepsilon/4)^2} \leq 1 + \varepsilon,$$

proving P1. Since $T(\Gamma') \leq T(\Gamma) = (1+\delta)L(\Pi)$, then

$$L(\Pi') \leq (1+\frac{\varepsilon}{8})\,T(\Gamma') \leq (1+\frac{\varepsilon}{8})(1+\delta)L(\Pi) \leq (1+\varepsilon)L(\Pi),$$

proving P2. P3 follows directly from that Γ' tracks Γ to a tolerance of (η_x, η_v), with $\eta_x = \varepsilon\rho/2$, and that Γ lies inside W.

Let i' and f' be the initial and final states of Γ', and X' and Y' be the initial and final states of Π'. Thus $\text{LOC}(X') = \text{LOC}(i')$, $\text{LOC}(Y') = \text{LOC}(f')$, $\text{VEC}(X') = \text{VEC}(i')/\|\text{VEC}(i')\|$ and $\text{VEC}(Y') = \text{VEC}(f')/\|\text{VEC}(f')\|$, and

$$
\begin{aligned}
&\|\text{VEC}(X') - \text{VEC}(X)\|_\infty \\
&\leq \|\text{VEC}(X') - \text{VEC}(X)\| \\
&= \|\text{VEC}(X') - \text{VEC}(i') + \text{VEC}(i') - \text{VEC}(i) + \text{VEC}(i) - \text{VEC}(X)\| \\
&\leq \|\text{VEC}(X') - \text{VEC}(i')\| + \|\text{VEC}(i') - \text{VEC}(i)\| + \|\text{VEC}(i) - \text{VEC}(X)\| \\
&\leq (1 - \|\text{VEC}(i')\|) + \sqrt{d}\eta_v + \frac{\delta}{1+\delta} \\
&\leq \frac{\varepsilon}{4} + \frac{\varepsilon}{8} + \frac{\varepsilon}{8} \\
&= \frac{\varepsilon}{2}.
\end{aligned}
$$

Similarly we can show that $\|\text{VEC}(Y') - \text{VEC}(Y)\| \leq \varepsilon/2$. This proves P4 for Γ'.

Such a trajectory Γ' (and thus a path Π') can be obtained by the TC-graph method. In constructing the graph, an edge is added if i) its a (μ, τ)-bang; ii) the bang does not diverge from W by more than $\varepsilon\rho/2$; and iii) the L_2 norms of the velocities of the bang fall into $[1-\varepsilon/4, 1+\varepsilon/8]$. Since μ and τ are $O(\varepsilon)$, the number of edges in this graph is bounded by $O(L^d(1/\varepsilon)^{6d-1})$, and thus the running time. This completes the proof of the lemma. □

Notice that we do not explicitly constrain the velocities of the tracking trajectory Γ'. This allows us to apply the Tracking Lemma. However, it happens that $\|v_{\Gamma'}(t)\|$ falls into a small range of $[1-\varepsilon/4, 1+\varepsilon/8]$, by being able

to track closely a trajectory Γ whose velocities are fixed to be 1 in L_2 norm. By upper bounding the accelerations and lower bounding the velocities of Γ', we are able to bound the curvature of Π'.

2.6.2 Approximating with obstacles

Let W be a d-dimensional configuration space of size L, with a set Ω of obstacles. Let $\mathcal{B}$ be the box decomposition of W, as described in Section 2.3.1. Let Π be an optimal $3l$-safe 1-constrained path from X to Y, where $\|\text{VEC}(X)\| = \|\text{VEC}(Y)\| = 1$.

As we did in the proof of the Safe Loose Tracking Lemma, we can divide Π into segments of paths, $\Pi_{U_0U_1} \| \ldots \| \Pi_{U_{m-1}U_m}$, such that $U_0 = X$, $U_m = Y$, and each U_i is the state where Π first exits $\xi(U_{i-1})$, for $1 \leq i \leq m$. Each segment $\Pi_{U_iU_{i+1}}$ is a 1-constrained path from U_i to U_{i+1} and lying inside $\xi(U_i)$. These are the paths we need to approximate.

We define the set of CS-grid states, the canonical extended boxes, the canonical starting states and the canonical ending states in the same way as we did in the previous sections, except that we let $\mathcal{V}$ be the set of *unit* vectors that are uniformly spaced with spacing δ. If $\delta = O(\epsilon)$ is chosen small enough, for any unit vector v, there is a unit vector $v' \in \mathcal{V}$ such that $\|v' - v\|_\infty \leq \epsilon/2$. Thus $\mathcal{V}$ suffices for the curvature-constrained case, since $\|\text{VEC}(U_i)\| = 1$ for all U_i's along Π. The following corollary can be derived from the Correcting Lemma.

Corollary 2.6.2 *Fix a constant $c > 1$, and let $\rho_c = 1/(c\kappa)$, where $\kappa > 0$ is arbitrary. Let Π be a κ-constrained path from state I to state F. Given any*

$\rho > 0$ and $\varepsilon > 0$, and given any two states U and V, if $\rho > \rho_c$, $\|\text{LOC}(F) - \text{LOC}(I)\|_\infty \geq \rho$, $U \in ngb(I, \varepsilon\rho/2, \varepsilon/2)$, and $V \in ngb(F, \varepsilon\rho/2, \varepsilon/2)$, we can construct a path Π' from U to V, by correcting Π, such that Π' satisfies the following properties:

P1. $L(\Pi') = (1 + e\varepsilon)L(\Pi)$,

P2. Π' *is* $((1 + e\varepsilon)^2/(1 - e\varepsilon)^2)\kappa$*-constrained,*

P3. $d(\Pi', \Pi) \leq (17/16)\varepsilon\rho$,

where $e = 4c\sqrt{d}$.

The consequence of Lemma 2.6.1 and Corollary 2.6.2 is that we are able to pre-compute paths which approximate U_{ii+1}'s and store them in the connection tables. Similar to what we did for the kinodynamic case, we can prove a version of the Loose Tracking Lemma and the Safe Loose Tracking Lemma for the curvature-constrained case. Since $|\mathcal{V}| = O((1/\epsilon)^{d-1})$ (as opposed to $O((1/\epsilon)^d)$ for the kinodynamic case), the number of edges in the CS-graph is reduced to $O((1/\epsilon)^{4d-4})$.

Combining the above, and choosing small enough $\epsilon = O(\varepsilon)$, we can obtain

Theorem 2.6.3 *Fix an $l > 0$ and an $\varepsilon > 0$. After some pre-computation, we can achieve the following.*

Let W be a d-dimensional space of size L and let Ω be a set of obstacles with a total of n vertices. Given any two states X and Y (with $\|\text{VEC}(X)\| = \|\text{VEC}(Y)\| = 1$), we can compute a collision-free $(1+\varepsilon)$-constrained path, from

$X' \in ngb(X, \varepsilon l, \varepsilon)$ to $Y' \in ngb(Y, \varepsilon l, \varepsilon)$, such that the path length is at most $(1+\varepsilon)$ times the length of an optimal 1-constrained $3l$-safe path from X to Y.

The running time of our algorithm is $O(nN + N \log N(1/\epsilon)^{4d-4})$, where $N = O((L/l)^d)$.

2.7 Correcting a Trajectory

Let Γ be an $(\bar{a}, \bar{v})$-trajectory from a state i to a state f. Given another pair of states g and h, we can construct a trajectory Γ' from g to h, by correcting Γ. Furthermore, we show that if g and h are close to i and f respectively, the correction is small.

For simplicity of illustration and analysis, we first look at the one dimensional case. Let $\delta a(t)$ be the corrective acceleration, i.e., $a_{\Gamma'}(t) = a_\Gamma(t) + \delta a(t)$, for $0 \leq t \leq T(\Gamma)$. Let $\delta v(t) = v_{\Gamma'}(t) - v_\Gamma(t)$, and $\delta p(t) = p_{\Gamma'}(t) - p_\Gamma(t)$. Also let $\Delta v_i = v_{\Gamma'}(0) - v_\Gamma(0) = \text{VEC}(g) - \text{VEC}(i)$, $\Delta v_f = v_{\Gamma'}(T) - v_\Gamma(T) = \text{VEC}(h) - \text{VEC}(f)$, $\Delta p_i = p_{\Gamma'}(0) - p_\Gamma(0) = \text{LOC}(g) - \text{LOC}(i)$, and $\Delta p_f = p_{\Gamma'}(T) - p_\Gamma(T) = \text{LOC}(h) - \text{LOC}(f)$. Thus

$$\begin{aligned}\delta v(t) &= v_{\Gamma'}(0) + \int_0^t (a_\Gamma(\mu) + \delta a(\mu))\, d\mu - (v_\Gamma(0) + \int_0^t a_\Gamma(\mu)\, d\mu) \\ &= \Delta v_i + \int_0^t \delta a(\mu)\, d\mu,\end{aligned}$$

and

$$\begin{aligned}\delta p(t) &= p_{\Gamma'}(0) + \int_0^t (v_\Gamma(\mu) + \delta v(\mu))\, d\mu - (p_\Gamma(0) + \int_0^t v_\Gamma(\mu)\, d\mu) \\ &= \Delta p_i + \int_0^t \delta v(\mu)\, d\mu \\ &= \Delta p_i + \int_0^t (\Delta v_i + \int_0^\mu \delta a(\nu)\, d\nu)\, d\mu \\ &= \Delta p_i + \Delta v_i t + \int_0^t \int_0^\mu \delta a(\nu)\, d\nu\, d\mu.\end{aligned}$$

Our object is to find $\delta a(t)$, such that

$$\Delta v_i + \int_0^T \delta a(\mu)\, d\mu = \Delta v_f, \text{ and}$$

$$\Delta p_i + \Delta v_i T + \int_0^T \int_0^\mu \delta a(\nu)\, d\nu\, d\mu = \Delta p_f,$$

where $T = T(\Gamma)$. We also want to keep the absolute values of $\delta v(t)$, $\delta a(t)$ and $\delta p(t)$ small.

Fix a constant $c \geq 1$ and let

$$\rho_c = \bar{v}^2/(c\bar{a}).$$

Assume that

$$|\text{LOC}(f) - \text{LOC}(i)| \geq \rho \geq \rho_c,$$

for some ρ, and let

$$\phi = \rho/\bar{v}.$$

Thus $T(\Gamma) \geq \phi$. Our correcting scheme is illustrated in Figure 2.4. Basically, we correct in three phases. In the first phase, we use a constant corrective

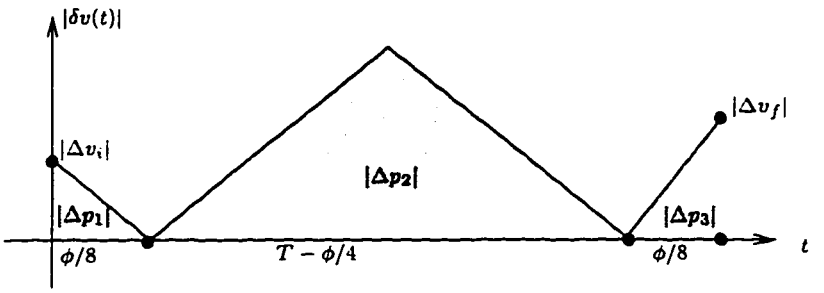

Figure 2.4: The correcting scheme.

acceleration to make $\delta v(t)$ become 0, while in the last phase, we use a constant corrective acceleration to make $\delta v(t)$ become Δv_f. The middle phase is used to correct the distance. Notice that at the beginning and the end of the second phase, $\delta v(t) = 0$.

The time length of the first and the last phase is $\phi/8$. Let Δa_1 be the constant corrective acceleration used in the first phase. Then

$$\Delta a_1 = -\frac{\Delta v_i}{\phi/8} = -\frac{8\Delta v_i}{\phi}.$$

If Δp_1 is the distance covered in this phase, then $\Delta p_1 = \Delta v_i \phi/16$. In this phase, $|\delta v(t)|$ is getting smaller and is upper bounded by $|\Delta v_i|$, while $|\delta p(t)|$ is upper bounded by $|\Delta p_i| + |\Delta p_1|$. Let Δa_3 be the constant corrective acceleration used in the last phase. Similarly, we have

$$\Delta a_3 = \frac{8\Delta v_f}{\phi}.$$

$\Delta p_3 = \Delta v_f \phi/16$ is the distance covered in this phase. In this phase, $|\delta v(t)|$ is upper bounded by $|\Delta v_f|$. At the beginning of this phase, $\delta p(t) = \Delta p_f - \Delta p_3$. Thus $|\delta p(t)|$ is upper bounded $|\Delta p_f| + |\Delta p_3|$ during this phase.

Let $T' = T(\Gamma) - \phi/4$ be the time spent in the second phase. This phase is divided into two sub-phases of equal length. The corrective accelerations used in these two sub-phases have the same absolute value, but opposite directions. The effect is that at the end of this phase, $\delta v(t)$ becomes 0 again. If Δp_2 is the distance covered in this phase, $\Delta p_2 = \Delta p_f - \Delta p_i - \Delta p_1 - \Delta p_3$. Notice that $|\Delta p_2|$ and $\delta p(t)$ are both upper bounded by $|\Delta p_i| + |\Delta p_1| + |\Delta p_3| + |\Delta p_f|$. Let Δa_2 be the corrective acceleration used in the first sub-phase. Then

$$\Delta a_2 = \frac{4\Delta p_2}{(T')^2}.$$

In this phase, $|\delta v(t)|$ is upper bounded by $2|\Delta p_2|/T'$.

Lemma 2.7.1 *For any $\varepsilon > 0$, and any two states g and h, if $g \in ngb(i, \varepsilon\rho/2, \varepsilon\bar{v}/2)$, and $h \in ngb(f, \varepsilon\rho/2, \varepsilon\bar{v}/2)$, where ρ is defined as above, then the trajectory Γ', constructed as above, satisfies the following properties:*

P1. $T(\Gamma') = T(\Gamma)$.

P2. Γ' *is a* $((1 + 4c\varepsilon)^2\bar{a}, (1 + 4c\varepsilon)\bar{v})$*-trajectory.*

P3. $d(\Gamma', \Gamma) \leq (17/16)\varepsilon\rho$.

Proof: P1 holds obviously. During the whole time period $T(\Gamma)$, the maximum

difference in position is upper bounded by

$$\begin{aligned}
|\Delta p_i| + |\Delta p_1| + |\Delta p_3| + |\Delta p_f| &\leq \frac{1}{2}\varepsilon\rho + \frac{|\Delta v_i|\phi}{16} + \frac{|\Delta v_f|\phi}{16} + \frac{1}{2}\varepsilon\rho \\
&\leq \varepsilon\rho + \frac{(\varepsilon\bar{v})(\rho/\bar{v})}{16} \\
&\leq \left(1 + \frac{1}{16}\right)\varepsilon\rho \\
&\leq \frac{17}{16}\varepsilon\rho.
\end{aligned}$$

This implies that $|p_{\Gamma'}(t) - p_{\Gamma}(t)| \leq (17/16)\varepsilon\rho$, for $0 \leq t \leq T(\Gamma)$. By definition, $d(\Gamma', \Gamma) \leq (17/16)\varepsilon\rho$, proving P3. The maximum difference in velocity is upper bounded by

$$\begin{aligned}
\max\left(|\Delta v_i|, |\Delta v_f|, \frac{2|\Delta p_2|}{T'}\right) &\leq \max\left(\frac{\varepsilon\bar{v}}{2}, \frac{2(17/16)\varepsilon\rho}{(1-1/4)\phi}\right) \\
&\leq \max\left(\frac{\varepsilon\bar{v}}{2}, \frac{3\varepsilon\rho}{\phi}\right) \\
&\leq \max\left(\frac{\varepsilon\bar{v}}{2}, 3\varepsilon\bar{v}\right) \\
&\leq 4c\varepsilon\bar{v}.
\end{aligned}$$

The maximum difference in acceleration is upper bounded by

$$\begin{aligned}
\max(|\Delta a_1|, |\Delta a_2|, |\Delta a_3|) &\leq \max\left(\frac{8|\Delta v_i|}{\phi}, \frac{4|\Delta p_2|}{(T')^2}, \frac{8|\Delta v_f|}{\phi}\right) \\
&\leq \max\left(\frac{4\varepsilon\bar{v}}{\phi}, \frac{4(17/16)\varepsilon\rho}{((3/4)\phi)^2}\right) \\
&\leq \max\left(\frac{4\varepsilon\bar{v}}{\phi}, \frac{8\varepsilon\bar{v}}{\phi}\right) \\
&= \frac{8\varepsilon\bar{v}}{\phi} \\
&= \frac{8\varepsilon\bar{v}^2}{\rho} \\
&\leq \frac{8\varepsilon\bar{v}^2}{\rho_c} \\
&= 8c\varepsilon\bar{a}.
\end{aligned}$$

Thus the velocity of Γ' is upper bounded by $(1 + 4c\varepsilon)\bar{v}$ and the acceleration is bounded by $(1 + 4c\varepsilon)^2\bar{a}$, proving P2. □

In higher dimensions, we can add corrective accelerations for each dimension separately. Since the time length of the trajectory is not changed, the corrections can be carried out simultaneously without affecting each other. Let $v_\Gamma^j(t)$ be $v_\Gamma(t)$ projected in the jth dimension, for $1 \leq j \leq d$. By the above lemma, we have

$$|v_{\Gamma'}^j(t)| \leq 4c\varepsilon\bar{v} + |v_\Gamma^j(t)|.$$

By triangle inequality,

$$\begin{aligned}
\|v_{\Gamma'}(t)\| &\leq \|v_{\Gamma}(t)\| + \|v_{\Gamma'}(t) - v_{\Gamma}(t)\| \\
&\leq \bar{v} + \sqrt{\sum_{j=1}^{d} (v^j_{\Gamma'}(t) - v^j_{\Gamma}(t))^2} \\
&\leq \bar{v} + \sqrt{\sum_{j=1}^{d} (4c\varepsilon\bar{v})^2} \\
&\leq \bar{v} + 4c\sqrt{d}\varepsilon\bar{v} \\
&= (1 + 4c\sqrt{d}\varepsilon)\bar{v}.
\end{aligned}$$

Similarly for the acceleration, we can obtain that

$$\|a_{\Gamma'}(t)\| \leq (1 + 4c\sqrt{d}\varepsilon)^2\bar{a}.$$

In summary,

Lemma 2.7.2 (Correcting Lemma) *Fix a constant $c \geq 1$, and let $\rho_c = \bar{v}^2/(c\bar{a})$, where $\bar{a}, \bar{v} > 0$ are arbitrary. Let Γ be an $(\bar{a}, \bar{v})$-trajectory from a state i to a state f. Given any $\rho > 0$ and $\varepsilon > 0$, and given any two states g and h, if $\rho > \rho_c$, $\|\text{LOC}(f) - \text{LOC}(i)\|_\infty \geq \rho$, $g \in ngb(i, \varepsilon\rho/2, \varepsilon\bar{v}/2)$, and $h \in ngb(f, \varepsilon\rho/2, \varepsilon\bar{v}/2)$, then we can construct a trajectory Γ' from g to h, by correcting Γ, such that Γ' satisfies the following properties:*

P1. $T(\Gamma') = T(\Gamma)$.

P2. Γ' *is a* $((1 + 4c\sqrt{d}\varepsilon)^2\bar{a}, (1 + 4c\sqrt{d}\varepsilon)\bar{v})$*-trajectory.*

P3. $d(\Gamma', \Gamma) \leq (17/16)\varepsilon\rho$.

2.8 Scaling Dynamic Bounds

Let Π and Π' be two paths. $\Pi' = \alpha\Pi$, or we say that Π' is Π scaled by a factor of α, if $L(\Pi') = \alpha\, L(\Pi)$ and $\Pi'(\ell) = \alpha\, \Pi(\ell/\alpha)$, for $0 \leq \ell \leq \alpha\, L(\Pi)$.

Lemma 2.8.1 (Velocity Scaling Lemma) *Let Γ be an $(\bar{a}, \bar{v})$-trajectory, and Π be its path. The path $(1/\alpha^2)\Pi$ can be traversed by an $(\bar{a}, \bar{v}/\alpha)$-trajectory Γ' in time $T(\Gamma)/\alpha$. Moreover Γ is an optimal $(\bar{a}, \bar{v})$-trajectory if and only if Γ' is an optimal $(\bar{a}, \bar{v}/\alpha)$-trajectory.*

Proof: Define Γ' to be the following: $p_{\Gamma'}(0) = p_\Gamma(0)/\alpha^2$, $v_{\Gamma'}(0) = v_\Gamma(0)/\alpha$ and $a_{\Gamma'}(t) = a_\Gamma(\alpha t)$, $0 \leq t \leq T(\Gamma)/\alpha$. We will show that, for $0 \leq t \leq T(\Gamma)/\alpha$,

$$v_{\Gamma'}(t) = v_\Gamma(\alpha t)/\alpha, \text{ and}$$

$$p_{\Gamma'}(t) = p_\Gamma(\alpha t)/\alpha^2.$$

This is because

$$\begin{aligned} v_{\Gamma'}(t) &= v_{\Gamma'}(0) + \int_0^t a_{\Gamma'}(s)\, ds \\ &= v_\Gamma(0)/\alpha + \int_0^t a_\Gamma(\alpha s)\, ds \\ &= v_\Gamma(0)/\alpha + \int_0^{\alpha t} a_\Gamma(s')(1/\alpha)\, ds' \\ &= v_\Gamma(0)/\alpha + (\int_0^{\alpha t} a_\Gamma(s')\, ds')/\alpha \\ &= v_\Gamma(\alpha t)/\alpha, \end{aligned}$$

and

$$\begin{aligned} p_{\Gamma'}(t) &= p_{\Gamma'}(0) + \int_0^t v_{\Gamma'}(s)\, ds \\ &= p_\Gamma(0)/\alpha^2 + \int_0^t v_\Gamma(\alpha s)/\alpha\, ds \\ &= p_\Gamma(0)/\alpha^2 + \int_0^{\alpha t} (v_\Gamma(s')/\alpha^2)\, ds' \\ &= p_\Gamma(0)/\alpha^2 + (1/\alpha^2)\int_0^{\alpha t} v_\Gamma(s')\, ds' \\ &= p_\Gamma(\alpha t)/\alpha^2. \end{aligned}$$

Let Π' be the path traversed by Γ'. Let ℓ and t be such that $\Pi'(\ell) = p_{\Gamma'}(t)$. Thus

$$\begin{aligned} \ell &= \int_0^t v_{\Gamma'}(s) ds \\ &= \int_0^t v_\Gamma(\alpha s)/\alpha ds \\ &= (1/\alpha^2)\int_0^{\alpha t} v_\Gamma(s) ds. \end{aligned}$$

This implies that $\Pi(\alpha^2\ell) = p_\Gamma(\alpha t)$. Since

$$\begin{aligned} \Pi'(\ell) &= p_{\Gamma'}(t) \\ &= p_\Gamma(\alpha t)/\alpha^2 \\ &= \Pi(\alpha^2\ell)/\alpha^2, \end{aligned}$$

therefore Π' is $(1/\alpha^2)\Pi$. The scaling preserves the optimality. This completes the proof of the lemma. □

Lemma 2.8.2 (Acceleration Scaling Lemma) *Let Γ be an $(\bar{a}, \bar{v})$-trajectory, and Π its path. The path $(1/\alpha)\Pi$ can be traversed by an $(\bar{a}/\alpha, \bar{v}/\alpha)$-trajectory*

Γ' *in time* $T(\Gamma)$. *Moreover* Γ *is an optimal* $(\bar{a}, \bar{v})$*-trajectory if and only if* Γ' *is an optimal* $(\bar{a}/\alpha, \bar{v}/\alpha)$*-trajectory.*

Proof: Define Γ' to be the following: $p_{\Gamma'}(0) = p_\Gamma(0)/\alpha$, $v_\Gamma(0) = v_\Gamma(0)/\alpha$ and $a_{\Gamma'}(t) = a_\Gamma(t)/\alpha$, $0 \leq t \leq T(\Gamma)$. We can show that, for any $0 \leq t \leq T(\Gamma)$,

$$v_{\Gamma'}(t) = v_\Gamma(t)/\alpha, \text{ and}$$

$$p_{\Gamma'}(t) = p_\Gamma(t)/\alpha.$$

This is because

$$\begin{aligned} v_{\Gamma'}(t) &= v_{\Gamma'}(0) + \int_0^t a_{\Gamma'}(s)\, ds \\ &= v_\Gamma(0)/\alpha + \int_0^t a_\Gamma(s)/\alpha\, ds \\ &= v_\Gamma(0)/\alpha + (\int_0^t a_\Gamma(s)\, ds)/\alpha \\ &= v_\Gamma(t)/\alpha, \end{aligned}$$

and

$$\begin{aligned} p_{\Gamma'}(t) &= p_{\Gamma'}(0) + \int_0^t v_{\Gamma'}(s)\, ds \\ &= p_\Gamma(0)/\alpha + \int_0^t v_\Gamma(s)/\alpha\, ds \\ &= p_\Gamma(0)/\alpha + (\int_0^t v_\Gamma(s)\, ds)/\alpha \\ &= p_\Gamma(t)/\alpha. \end{aligned}$$

Let Π' be the path traversed by Γ'. Similar to the proof of the previous lemma, we can show that $\Pi' = (1/\alpha)\Pi$. This completes the proof of the lemma. □

Lemma 2.8.3 (Acceleration-Velocity Scaling Lemma) *Let Γ be an $(\bar{a}, \bar{v})$-trajectory, and let Π be its path. The path $(\beta/\alpha^2)\Pi$ can be traversed by an $(\bar{a}/\beta, \bar{v}/\alpha)$-trajectory Γ' in time $(\beta/\alpha)\,T(\Gamma)$. Moreover Γ is an optimal $(\bar{a}, \bar{v})$-trajectory if and only if Γ' is an optimal $(\bar{a}/\beta, \bar{v}/\alpha)$-trajectory.*

Proof: Let $\Pi'' = (1/\beta)\Pi$. By Lemma 2.8.2, Π'' can be traversed by an $(\bar{a}/\beta, \bar{v}/\beta)$-trajectory Γ'' in time $T(\Gamma)$.

Let $\gamma = \alpha/\beta$, and let $\Pi' = (1/\gamma^2)\Pi''$. By Lemma 2.8.1, Π' can be traversed by an $(\bar{a}/\beta, \bar{v}/(\beta\gamma))$-trajectory in time $T(\Gamma'')/\gamma$. Since

$$\Pi' = (1/\gamma^2)\Pi'' = (1/\gamma^2)(1/\beta)\Pi = (\beta/\alpha^2)\Pi,$$

$$\bar{v}/(\beta\gamma) = \bar{v}/\alpha,$$

and

$$T(\Gamma'')/\gamma = T(\Gamma)/\gamma = (\beta/\alpha)T(\Gamma),$$

therefore, this trajectory is the one desired. □

Corollary 2.8.4 *Given a kinodynamic motion-planning problem with the following parameters: the world size L, the dynamic bounds $\bar{a}$ and $\bar{v}$, the initial state i and the final state f, we can scale the problem to one with $\bar{a} = 1$ and $\bar{v} = 1$, by scaling the initial and final velocities by a factor of $(1/\bar{v})$ and scaling the locations of the world (including the obstacles and $\text{LOC}(i)$ and $\text{LOC}(f)$) by a factor of $(\bar{a}/\bar{v}^2)$. Let Γ be the trajectory obtained for the scaled problem, then the trajectory Γ' such that*

$$a_{\Gamma'}(t) = \bar{a} \cdot a_{\Gamma}((\bar{a}/\bar{v})t),$$

for $0 \leq t \leq (\bar{v}/\bar{a})T(\Gamma)$ is the solution for the original problem.

Chapter 3

Curvature-Constrained Shortest Paths

3.1 Introduction

Let B be a point robot. A *position* X for B is a pair $(\text{LOC}(X), \text{VEC}(X))$, where $\text{LOC}(X)$ is a point representing the location of the robot and $\text{VEC}(X)$ is an angle between 0 and 2π, representing its orientation, both lying in the plane. A *path* is an oriented curve; $\|\cdot\|$ denote the length of a path. A path is called *legal* if its average curvature is at most 1 in every interval. Let Ω be a set of disjoint polygonal obstacles, with a total of n vertices. A legal path is *feasible* (with respect to Ω) if it does not intersect the interior of any obstacle of Ω. A path Π from a position X to another position Y is *optimal*, if it is a feasible path with the minimum arc length, where the minimum is taken over all feasible paths from X to Y (it can be shown that the minimum always exists). Finally, following the definition in [48], a path Π is called *robust* if it is feasible, and even after a small perturbation in orientation (resp. location) at each position where Π passes through an obstacle vertex (resp. edge), Π remains feasible and

passes through the same set of obstacle vertices and edges. Π is called *δ-robust* if a perturbation of up to $\pm\delta/2$ can be made in the orientation and location. Π is an *optimal δ-robust* path from X to Y if its length is the minimum over all δ-robust paths from X to Y.

The first main result of our paper is an efficient algorithm for computing an approximation to the optimal robust path (Section 3.3). More precisely, given an obstacle environment Ω with n vertices, two positions I and F, and a parameter ε, we present an $O((n^2/\varepsilon^4)\log n)$-time algorithm that computes a feasible path from I to F whose length is at most $(1+\varepsilon)$ times the length of an optimal ε-robust path from I to F. Compared with the Jacobs-Canny algorithm [48], our algorithm is not only considerably faster in terms of the complexity of Ω, but, more importantly, the running time is independent of L, the total edge length of Ω. The second improvement is rather significant, because one cannot assume L to be small. For unconstrained optimal path planning, one can scale down the environment arbitrarily (to reduce the value of L), compute a shortest path in the scaled environment, and then scale the optimal path back. But this scaling trick does not work for curvature-constrained shortest paths, as the scaling will change the curvature of the path as well. The path returned by our algorithm is not necessarily robust. We can, however, modify the algorithm to compute an $(\varepsilon/2)$-robust path in time $O((n^{2.5}/\varepsilon^4)\log n)$, whose length is no more than $(1+\varepsilon)$-times the length of an optimal ε-robust path. In fact, an $O(n^{2+\gamma}/\varepsilon^4)$-time algorithm can be obtained, using the recent range-searching data structures [1], but we will not discuss this improvement in this paper.

Our algorithms are based on a stronger characterization of optimal (and of optimal robust) paths, which is interesting in its own right (Section 3.2.1). Roughly speaking, we prove that, except in some very special cases, an optimal path touches an edge e of Ω only at points near a vertex v (v is not necessarily one of the endpoints of e), such that these points are visible from v. In other words, we can ignore the portions of the edges of Ω that are not near to any vertex of Ω, which enables us to develop an algorithm whose running time does not depend on the size of Ω.

3.2 Characterization of Optimal Paths

In this section we characterize optimal paths in presence of obstacles, in the plane. Let Π be a feasible path. We call a nonempty subpath of Π a *C-segment* (resp. *L-segment*) if it is a circular arc of unit radius (resp. line segment) and maximal. Suppose Π consists of a C-segment, L-segment, and a C-segment, then we will say that Π is of type CLC. This notion can be generalized to an arbitrarily long sequence. Dubins [36] proved the following result.

Theorem 3.2.1 (Dubins [36]) *In an obstacle-free environment, an optimal path between any two positions is of type CCC or CLC, or a substring thereof.*

We will refer to such paths as *Dubins paths*. In the presence of obstacles, Fortune and Wilfong [39] and Jacobs and Canny [48] observed that any subpath of an optimal path that does not touch any obstacle except at the endpoints is a Dubins path. In particular,

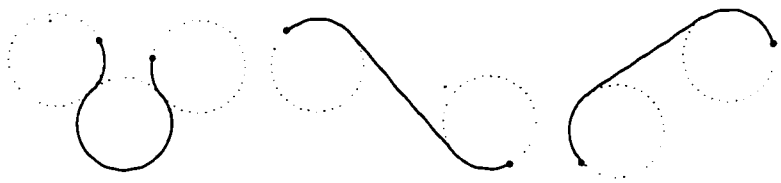

Figure 3.1: Examples of Dubins paths.

Theorem 3.2.2 (Fortune-Wilfong [39], Jacobs-Canny [48]) *Given an obstacle environment Ω, an initial position I, and a final position F, an optimal path from I to F consists of a sequence $\Pi_1||\cdots||\Pi_k$ of feasible paths, where each Π_i is a Dubins path from a position X_{i-1} to a position X_i, such that $X_0 = I$, $X_k = F$, and, for $0 < i < k$, $\text{LOC}(X_i) \in \partial\Omega$.*

Although Jacobs and Canny [48] did not prove it explicitly, the above theorem holds even for δ-robust optimal paths. (Their approximation algorithm uses this stronger claim implicitly.) This theorem implies that an optimal path is a finite sequence of C- and L-segments, so it can be represented as a finite string over the alphabet $\{C, L\}$. There are, however, infinite number of such paths — an optimal path can touch a vertex at an arbitrary orientation, or it can touch an arbitrary point of an edge. Jacobs and Canny [48] observe that if one is interested only in computing $(1+\varepsilon)$-approximate paths, one can choose a finite set of orientations at which a path can touch a vertex, and can also choose a finite set of points on each edge e at which a path can touch e. The number of points chosen on an edge is proportional to the length of the edge. That is why the running time of their algorithm depends on the total length of edges of Ω. We circumvent this problem by proving a stronger property of

optimal paths.

A feasible C-segment is called *free* if it does not intersect Ω; *anchored* if it touches $\partial\Omega$ at two or more points; and *semi-free* if it touches $\partial\Omega$ at exactly one point. By Theorems 3.2.1 and 3.2.2, an optimal path cannot have two consecutive free C-segments. Moreover, there are only finite number of circles that touch $\partial\Omega$ at two or more points (assuming that there are no two edges parallel at distance 1), so there are only finite number of circles that may contain anchored C-segments. We thus need a better understanding of semi-free C-segments.

The following theorem states the main result of this section.

Theorem 3.2.3 *If w is a semi-free C-segment of an optimal path, then there is a vertex v of Ω within distance* 15 *from the intersection point ξ of w and $\partial\Omega$ that is visible from ξ (i.e., the interior of the segment ξv does not intersect Ω).*

3.2.1 Proof of Theorem 3.2.3

We will prove the theorem by a sequence of lemmata. We begin with a few notation and simple observations. For a circle C and two points $a, b \in C$, let $C[a, b]$ denote the arc of C from a to b in the clockwise direction. For an oriented path Π and two points $a, b \in \Pi$, let $\Pi[a, b]$ denote the subpath of Π from a to b.

If ξ is a vertex of Ω, then there is nothing to prove, so assume that ξ lies in the interior of an edge $e = pq$. Without loss of generality, assume that e lies on the x-axis, that w lies below the x-axis, and that w is oriented clockwise. We

can also assume that $d(p,\xi), d(q,\xi) > 15$. We will regard $\text{LOC}(I)$ and $\text{LOC}(F)$ as the vertices of Ω.

We divide the proof of Theorem 3.2.3 into several cases. For each case, we prove the existence of a closed, simply-connected region R satisfying the following properties:

P1. R lies on or below the x-axis, R contains ξ, and either ∂R contains a vertex of Ω or the interior of R contains a point of $\partial\Omega$.

P2. If an edge $g \in \Omega$ intersects the interior of R, it crosses ∂R in at most one point, which implies that at least one of the endpoints of g lies inside R.

P3. $d(\xi, x) \leq 15$ for all points $x \in R$.

See Figure 3.2 for an example. P1–P3 together imply that there is a vertex of Ω within distance 15 from ξ. In order to prove that there is also a vertex within distance 15 from ξ which is visible from ξ, we introduce R^*, the convex hull of R.

Lemma 3.2.4 *If a simply connected region R satisfies P1–P3, then R^* also satisfies P1–P3.*

Proof: P1 is obvious. If an obstacle edge g crosses ∂R^* at two points, then, using a continuity argument and the fact that R is simply connected, one can prove that g also crosses ∂R at two points, which contradicts property P2 of R. Hence, g crosses ∂R^* in at most one point, thereby proving P2.

For any point $x \in R^*$, the ray emanating from ξ and passing through x intersects ∂R^* at one point, say y (y may be identical to x). If $y \in R$, property

P3 follows since $d(\xi, x) \leq d(\xi, y) \leq 15$. Otherwise y lies in the interior of a line segment, both of whose endpoints, say u and v, lie on ∂R. Property P3 also follows since $d(\xi, x) \leq d(\xi, y) \leq \max\{d(\xi, u), d(\xi, v)\} \leq 15$. This completes the proof of the lemma. □

For each obstacle edge e that intersects R^*, clip it within R^*. Let E denote the set of clipped segments. (If an endpoint of an edge lies on ∂R^*, but the edge does not intersect the interior of R^*, we add that endpoint as a degenerate segment to E.) By property P1, $E \neq \emptyset$. We define the following partial ordering on the segments of E. We say that $e_i \prec e_j$ $(e_i, e_j \in E)$ if any ray emanating from ξ and intersecting both e_i and e_j intersects e_i before intersecting e_j (That is, viewed from ξ, e_j cannot occlude any portion of e_i). Such an ordering always exists because the segments of E are disjoint, they lie below the x-axis, and ξ lies on the x-axis; see, e.g., [25, 42]. Moreover, this partial ordering can be extended to a total ordering. By the definition of the ordering, every point on the first segment in this ordering is visible from ξ. But one of the endpoints of this segment, say, v, is a vertex of Ω, so we have found an obstacle vertex v within distance 15 from ξ that is visible from ξ.

The following simple lemma, which will be useful in many cases, follows from Lemma 3.2.4.

Lemma 3.2.5 *Let α and β be two points on an optimal path Π from I to F such that α, ξ, and β appear in that order along Π, such that $\Pi[\alpha, \beta]$ lies below or on the x-axis, and such that $d(\xi, x) \leq 15$ for all $x \in \Pi[\alpha, \beta]$. Let R be the convex hull of $\Pi[\alpha, \beta]$. If α is an obstacle vertex or if it lies in the interior of*

an obstacle edge whose supporting line intersects $\Pi[\alpha, \beta]$ *at a point other than* α*, then* R *satisfies P1-P3. See Figure 3.2.*

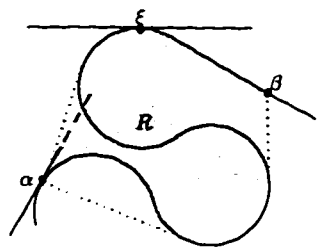

Figure 3.2: $\Pi[\alpha, \beta]$ and its convex hull.

Proof: Since R is the convex hull of $\Pi[\alpha, \beta]$, R is a simply connected region. We can show that R satisfies P2 and P3, following an argument similar to the one in the proof of Lemma 3.2.4. If α is a vertex, R satisfies P1. Otherwise the obstacle edge containing α lies in the interior of R near α, because the line supporting this edge intersects the interior of R. Hence, R satisfies P1 also. This completes the proof. □

Since we regard LOC(I) and LOC(F) as vertices of Ω and assume that ξ is not a vertex, the semi-free C-segment w has to be a middle segment of Π. Let w^- (resp. w^+) be the segment of Π immediately before (resp. after) w.

Using a perturbation argument, similar to the one used in [2] (see Figure 3.3), one can easily prove the following lemma, whose proof is omitted from here.

Lemma 3.2.6 *If* w *is neither the first segment nor the last segment of* Π*, then (i)* $\|w\| > \pi$*, and (ii) either* w^- *or* w^+ *is a C-segment.*

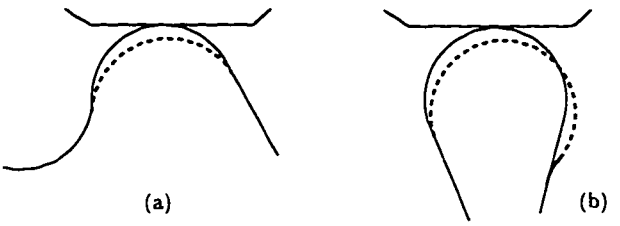

Figure 3.3: Length reducing perturbations: (a) $\|w\| \leq \pi$, (ii) both w^- and w^+ are L-segments.

Let us assume that w^- is a C-segment. Let ξ^- (resp. ξ^+) be the point on Π that intersects Ω immediately before (resp. after) ξ. Since $\Pi[\xi^-, \xi]$ is a Dubins path and w^- is a C-segment, $\Pi[\xi^-, \xi]$ consists of two or three C-segments. In either case,

$$d(\xi, x) \leq 6, \tag{3.1}$$

for any $x \in \Pi[\xi^-, \xi]$. Here onwards, we will assume that

($\star$) ξ^- lies in the interior of an edge e^- and the line, ℓ^-, supporting e^- does not intersect $\Pi[\xi^-, \xi]$.

If Π does not satisfies ($\star$), Theorem 3.2.3 easily follows from Lemma 3.2.5. Let σ be the intersection point of ℓ^- and the x-axis. Without loss of generality, assume that σ lies to the left of ξ.

We consider the following three cases and prove the existence of a region R satisfying P1–P3 for each case separately.

Case 1. The angle $\angle\xi\sigma\xi^- \geq \pi/6$; see Figure 3.4.

Case 2. The angle $\angle\xi\sigma\xi^- < \pi/6$.

Case 2.1. $d(\xi, \xi^+) < 8$; see Figure 3.5.

Case 2.2. $d(\xi, \xi^+) \geq 8$; see Figure 3.6.

Lemma 3.2.7 *There exists a region R satisfying P1-P3, for Case 1, i.e., when $\angle\xi\sigma\xi^- \geq \pi/6$.*

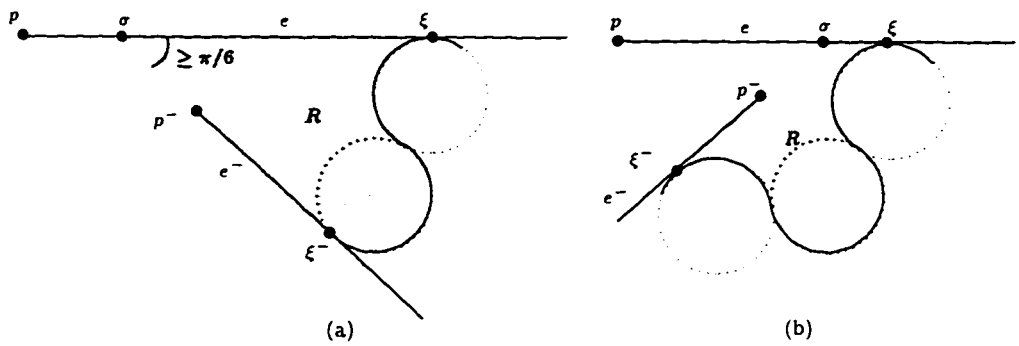

Figure 3.4: $\angle\xi\sigma\xi^- \geq \pi/6$: (a) $\Pi[\xi^-, \xi]$ has two C-segments; (b) $\Pi[\xi^-, \xi]$ has three C-segments.

Proof: We define R to be the closed region bounded by $\Pi[\xi^-, \xi]$ and the segments $\sigma\xi$ and $\sigma\xi^-$. See Figure 3.4. By assumption $(\star)$, neither x-axis nor ℓ^- intersects the interior of $\Pi[\xi^-, \xi]$, therefore R is a simply connected region. Using the sine law and the fact that $d(\xi, \xi^-) \leq 6$ (see (3.1)), we obtain

$$d(\xi, \sigma) = d(\xi, \xi^-)\frac{\sin\angle\sigma\xi^-\xi}{\sin\angle\xi\sigma\xi^-} \leq \frac{d(\xi, \xi^-)}{\sin(\pi/6)} \leq \frac{6}{\sin(\pi/6)} \leq 12.$$

Since $d(p, \xi) > 15$, we have $\sigma \in e$. The obstacle edges are disjoint, so the left endpoint of e^-, say p^-, has to lie on the segment $\sigma\xi^-$, and thus on the boundary of R, implying that R satisfies P1. R also satisfies P2 because any obstacle edge $g \in \Omega$ can cross ∂R only at the segment σp^-. Finally, for any $x \in R$, $d(\xi, x) \leq \max\{d(\xi, \sigma), 6\} \leq 12$, therefore R satisfies P3 as well. □

Lemma 3.2.8 *There exists a region R satisfying P1-P3, for Case 2.1, i.e., when $\angle\xi\sigma\xi^- < \pi/6$ and $d(\xi, \xi^+) < 8$.*

Proof: If ξ^+ is an obstacle vertex, then the claim follows from Lemma 3.2.5, so assume that ξ^+ lies in the interior of an obstacle edge e^+. Since $d(\xi, \xi^+) \leq 8$, ξ^+ lies in a disc D_ξ of radius 8 centered at ξ. Let u (resp. v) be the point on e at distance 8 from ξ to its left (resp. right). If the left endpoint of e^- lies to the left of u, let u' be the point on e^- with the same x-coordinate as u; otherwise, let u' be the left endpoint of e^-. Similarly, if the right endpoint of e^- lies to the right of v, let v' be the point on e^- with the same x-coordinate as v; otherwise let v' be the right endpoint of e^-. See Figure 3.5. We set R to be the quadrilateral $uu'v'v$. R obviously satisfies P1, because either an endpoint of e^- lies on ∂R (see Figure 3.5b for an example), or ξ^+ (which is a point on the edge e^+) lies in the interior of R (this is because by the construction of R, R contains the portion of the disc D_ξ in between e and e^-; see Figure 3.5a for an example). An obstacle edge g cannot intersect uv or $u'v'$ (as they are portions of obstacle edges), and g cannot intersect both uu' and vv' (as this would imply that g intersects $\Pi[\xi^-, \xi]$). Hence R satisfies P2.

Finally, we prove that R satisfies (P3). Let v'' be the point on ℓ^- that has the same x-coordinate as v ($v'' = v'$ if the right endpoint of e^- lies to the right of v). It can be shown that, for any $x \in R$, $d(\xi, x) \leq d(\xi, v'')$ (here we are assuming that ℓ and ℓ^- intersect to the left of ξ). Let s (resp. s') be a point on e^-, such that $\xi s \perp e^-$ (resp. $\xi s' \perp e$). Let s'' be a point on the segment vv'' such that $s's'' \perp vv''$. Notice that $\angle s\xi s' = \angle v''s's'' = \angle\xi\sigma\xi^- < \pi/6$, where

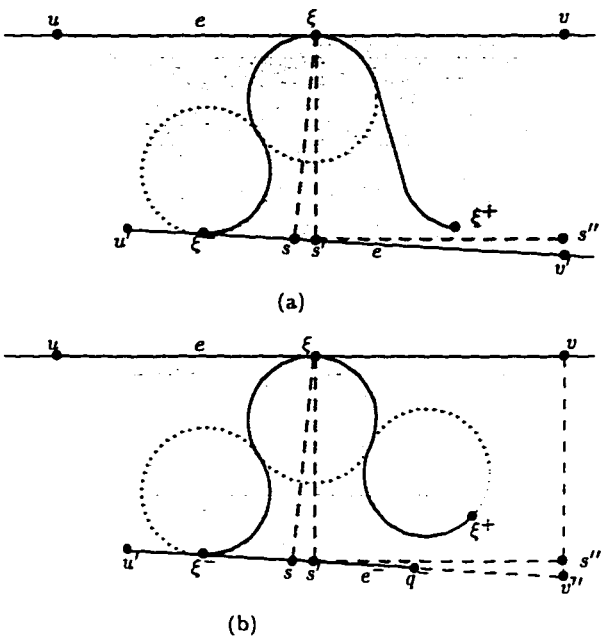

Figure 3.5: $\angle\xi\sigma\xi^+ < \pi/6$ and $d(\xi,\xi^+) < 8$: (a) ξ^+ lies in the interior of R; (b) q^- lies on ∂R.

σ is the intersection point of lines containing e and e^-. Since

$$d(v,s'') = d(\xi,s') = d(\xi,s)\sec\angle s\xi s' < d(\xi,\xi^-)\sec\frac{\pi}{6} \le 6\cdot\frac{2}{\sqrt{3}} = \frac{12}{\sqrt{3}},$$

and

$$d(s'',v'') = d(s',s'')\tan\angle v''s's'' < d(\xi,v)\tan\frac{\pi}{6} = 8\cdot\frac{1}{\sqrt{3}} = \frac{8}{\sqrt{3}},$$

we have $d(v,v'') < 20/\sqrt{3}$. Therefore for any $x \in R$,

$$d(\xi,x) \le d(\xi,v'') = \sqrt{d(\xi,v)^2 + d(v,v'')^2} \le \sqrt{8^2 + (20/\sqrt{3})^2} \le 15.$$

This completes the proof of the lemma. □

Lemma 3.2.9 *There exists a region R satisfying P1-P3, for Case 2.2, i.e., when $\angle\xi\sigma\xi^- < \pi/6$ and $d(\xi, \xi^+) \geq 8$.*

Proof: If w^+, the segment of Π following w, is a C-segment, then, by Theorem 3.2.1, $\Pi[\xi, \xi^+]$ consists of at most three C-segments and $d(\xi, \xi^+) \leq 6$, which contradicts the assumption that $d(\xi, \xi^+) > 8$. Hence w^+ is an L-segment, $\Pi[\xi, \xi^+]$ is of CLC type, and $\|w^+\| \geq d(\xi, \xi^+) - 4 > 4$.

Let C_1 and C_2 be the circles containing w^- and w, respectively, and let h be the line supporting w^+. There are two cases to consider:

(i) h does not intersect C_1,

(ii) h intersects C_1.

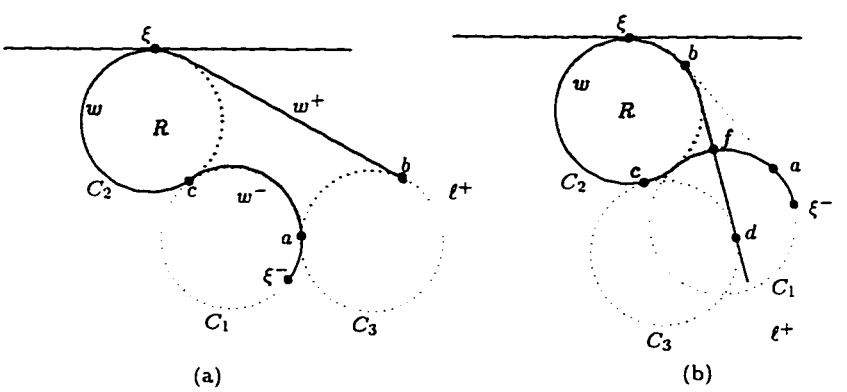

Figure 3.6: $\angle\xi\sigma\xi^- < \pi/6$ and $d(\xi, \xi^+) \geq 8$: (a) h does not intersect C_1; (b) h intersects C_1.

Case (i): See Figure 3.6a. Let C_3 be the other (unit-radius) circle tangent to both h and C_1. Let a (resp. b) be the intersection point of C_3 with C_1 (resp. h). Since $|w^+| > 4$, b lies on the segment w^+.

Let c be the common point of C_1 and C_2. Any line tangent to C_1 at a point on the arc $C_1[c,a]$ intersects the path $\Pi[c,b]$. Hence, by assumption $(\star)$, ξ^- can not lie on $C_1[c,a]$, i.e., $a \in \Pi[\xi^-,\xi]$. Define R to be the closed region bounded by $\Pi[a,b]$ and the segment ab; see the shaded region in Figure 3.6a. If the arc $C_3[a,b]$ does not cross any obstacle edge, we can shorten Π by replacing $\Pi[a,b]$ with $C_3[a,b]$, contradicting the optimality of Π. Hence, an edge of Ω intersects $C_3[a,b]$, thus, also intersects the interior of R; thereby implying that R satisfies P1. Since an edge of Ω can cross ∂R only at ab, P2 is obvious. Finally, every point on ∂R is within distance 15 from ξ, P3 also follows.

Case (ii): See Figure 3.6b. Let f be the first intersection point of h with C_1. Let ab be the segment tangent to C_1 and C_2 at a and b respectively. Let C_3 be the unit-radius circle tangent to C_2 and h at c and d respectively. Since $|w^+| > 4$, both d and f lie on the segment w^+.

The same argument as in case (i) implies that $a \in \Pi[\xi^-,\xi]$. Define R to be the region bounded by the segment ab, the circular arc $C_2[c,b]$ the segments cd and df, and the circular arc $C_1[f,a]$; see the shaded region in Figure 3.6b. An obstacle edge crosses either the segment ab or the arc $C_3[c,d]$, thus intersecting the interior of R, because otherwise the path obtained by concatenating the segment ab and the circular arcs $C_2[c,b]$ and $C_3[c,d]$ is shorter than the path $\Pi[a,d]$, contradicting the optimality of Π. This means that R satisfies P1. An obstacle edge can cross ∂R only at segments cd and ab, and none of the edges can cross both the segments (because then it would cross Π), P2 follows. Finally, every point on ∂R is within distance 15 from ξ, so P3 also follows. □

These lemmata together complete the proof of Theorem 3.2.3.

A closer look at the proof reveals that Theorem 3.2.3 holds even if Π is a δ-robust optimal path. As mentioned in the beginning of the section, Theorem 3.2.2 is true even for δ-robust optimal paths, therefore it suffices to argue that Lemmas 3.2.6–3.2.9 hold for the δ-robust case. Indeed, Lemmas 3.2.7 and 3.2.8 do not perform any perturbation and rely only on the feasibility of Π, so they obviously hold for robust paths also. Lemmas 3.2.6 and 3.2.9 perturb the path in such a way that the positions at which the perturbed path touches Ω is a subset of those at which the original path touches Ω (see e.g., Figure 3.6). Hence, by the definition of robustness, if Π is robust, then the perturbed path is also robust. We can therefore conclude

Theorem 3.2.10 *If w is a semi-free C-segment of an optimal δ-robust path, then there is a vertex v of Ω within distance* 15 *from the intersection point ξ of w and $\partial\Omega$ that is visible from ξ.*

3.2.2 Anchored C-segments

Recall that a C-segment is called anchored if it touches $\partial\Omega$ at two (or more) points. We prove a property of anchored C-segments, which will be useful later. We call an anchored C-segment, touching $\partial\Omega$ at two points $p_1 \in e_1$ and $p_2 \in e_2$, *good* if we can find two obstacle vertices v_1 and v_2 (not necessarily distinct) such that v_i is visible from p_i and $d(p_i, v_i) \leq 15$. Otherwise, it is called *bad*.

We first mention a simple lemma, which can be proved using a perturbation

argument similar to the one in Figure 3.3. We omit the proof from here, and refer the reader to [2].

Lemma 3.2.11 *An optimal does not contain a C-segment that touches two parallel obstacle edges at their interior points, and that does not touch any other obstacle edge.*

Lemma 3.2.12 *If w is a feasible bad anchored C-segment, then no obstacle edge intersects the interior of the circle containing w.*

Proof: Let w be an anchored C-segment touching $\partial\Omega$ at two points $p_1 \in e_1$ and $p_2 \in e_2$. Since w is a bad anchored C-segment, one of p_i, say p_1, has to be at least 15 distance away from the endpoints of e_i. Without loss of generality, let $e = e_1$, $e^- = e_2$, $\xi = p_1$, $\xi^- = p_2$, where e, e^-, ξ, ξ^- are as defined in the last section.

It can be shown that the angle between e and e^- is $< \pi/6$. Otherwise, following the proof of Lemma 3.2.7, we can find a vertex v_i, such that v_i is visible to p_i and $d(v_i, p_i) \leq 15$, contradicting the assumption that w is a bad anchored C-segment.

Let C be the circle containing w. We construct a region R, as in the proof of Lemma 3.2.8 (except that we set $d(\xi, u) = d(\xi, v) = 2$, since the distance between any point inside C and ξ is ≤ 2). R satisfies P2 (since w is feasible) and P3. If an obstacle edge intersects the interior of C, it also intersects R, making R satisfy P1. If so, we can find for each $i = 1, 2$ a vertex v_i such that v_i is visible from p_i and $d(v_i, p_i) \leq 15$, contradicting that w is a bad anchored

C-segment. Thus no obstacle edge intersects the interior of C. This proves the lemma. □

Lemma 3.2.13 *There are only $O(n)$ circles that can contain a feasible bad anchored C-segment.*

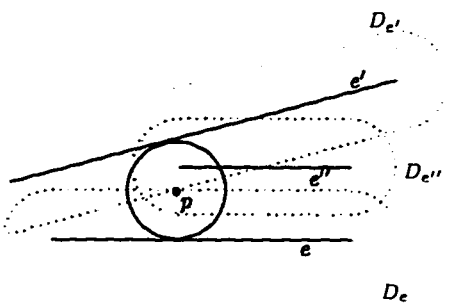

Figure 3.7: The straight line edges of D_e and $D_{e'}$ intersect at p. The unit radius circle centered at p is tangent to both e and e', and is crossed by e'', whose race-track $D_{e''}$ contains p in its interior.

Proof: For each edge $e \in \Omega$, let D_e be the Minkowski sum of e and the unit-radius disk. D_e is a race-track bounded by two semi-circles of unit radius and two translated copies of e; see Figure 3.7. An intersection point p of the straight-line edges of D_e and $D_{e'}$ corresponds to the center of a unit-radius circle tangent to e and e' at their interior points. (Notice that there may be an infinite number of intersection points if e and e' are parallel and unit distance apart.) A point p lies inside D_e if and only if e intersects the interior of the unit-radius circle centered at p. Set $\mathcal{D} = \bigcup_{e \in \Omega} D_e$. A unit-radius circle is tangent to two edges and does not intersect any obstacle edge in its interior, only if its center is at a vertex of $\partial\mathcal{D}$, or its center lies on a line segment of $\partial\mathcal{D}$, which is the common part of the race-tracks of two edges parallel and unit

distance apart. By Lemma 3.2.11, a feasible C-segment tangent to two parallel edges can not be bad, thus the number of circles that can contain a feasible bad anchored C-segment is at most the number of vertices of $\partial\mathcal{D}$, which, by a result of Kedem et al. [57], is $O(n)$; so the lemma follows. □

3.3 Computing Near Optimal Paths

In this section we present an efficient algorithm for computing a feasible path whose length is no more than $(1+\varepsilon)$ times the length of an optimal ε-robust path, for any $\varepsilon > 0$. As in [48], we construct a weighted directed graph $G = (V, E)$, where V is a set of discretized positions. These positions are obtained by discretizing the set of orientations at which a path touches a vertex and the set of points at which a path touches an edge. Jacobs and Canny [48] choose points uniformly spaced along each edge. We, on the other hand, use Theorem 3.2.3 and Lemma 3.2.13 to choose points more carefully, as described in Section 3.3.1. There is an edge $(X, Y) \in E$ from a position X to another position Y if there exists a feasible Dubins path from X to Y. If there is more than one such path, we choose the one with the minimum arc length. The weight of an edge is the arc-length of the chosen Dubins path. We claim that by choosing the proper parameter δ for discretization, for any optimal ε-robust path Π from X to Y, there is a graph path from X to Y in G whose length is at most $(1+\varepsilon)$ times the length of Π. The choice of δ and the proof of this claim are given in Section 3.4. Therefore, the problem reduces to computing a shortest path in G, which can be done in time $O(|V|^2)$, using Dijkstra's shortest-path algorithm.

3.3.1 Computing the node set

In this subsection we describe the node set V and an efficient algorithm for computing it. We set $V = \{I, F\} \cup V_1 \cup V_2 \cup V_3$, where each subset V_i corresponds to a specific type of positions. The first set V_1 corresponds to positions located at the vertices of Ω. More precisely, for each vertex v of Ω, V_1 contains the positions $(v, i\delta)$ for $0 \leq i \leq \lfloor 2\pi/\delta \rfloor$. V_1 can be constructed in $O(n/\delta)$ time in a straight-forward manner.

The second set V_2 corresponds to bad anchored C-segments. For a pair of non-parallel edges e_1, e_2, if the unit-radius circle C tangent to e_1 and e_2 does not intersect any other edge of Ω, we add four positions (p_1, θ_1), $(p_1, \theta_1 + \pi)$, (p_2, θ_2), and $(p_2, \theta_2 + \pi)$ to V_2, where p_i is the point at which C is tangent to e_i, and θ_i is the angle between e_i and the x-axis. Using an algorithm of Kedem et al. [57], the set of unit-radius circles tangent to two edges and not intersecting any other edge can be computed in $O(n \log^2 n)$ time, and so can be the set V_2.

The third set V_3 corresponds to semi-free C-segments and good anchored C-segments. For each edge $e \in \Omega$, we first mark a portion $\hat{e}$ of it, as described below, and then choose those points on e that are at distance $i\delta$ from its left endpoint, for any natural number i such that the semi-open interval $[i\delta, (i+1)\delta)$ intersects $\hat{e}$. Let S^e denote the set of points selected on e. Assuming that the angle between e and the x-axis is θ, for each point $p \in S^e$, we add two positions (p, θ) and $(p, \theta + \pi)$ to V_3.

We now describe which portion of each edge is marked. We mark the edges in two stages. First, for each edge e, we mark the portion that lies within

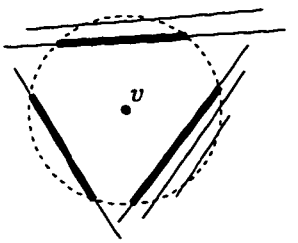

Figure 3.8: Segments in E_v; fat edges denote the portion of E_v visible from v.

distance 30 from any of its endpoints. Next, for each vertex $v \in \Omega$, let D_v be the disk of radius 15 centered at v. Let E_v be the set of edges that contain at least one unmarked point inside D_v, i.e., $e \in E_v$ if $e \cap D_v$ contains at least one point whose distance to both endpoints of e is more than 30. For each edge $e \in E_v$, we mark the portion of $e \cap D_v$ that is visible from v with respect to the edge set E_v (i.e., we ignore the edges in the set $E \setminus E_v$). We repeat this step for all vertices of Ω.

Lemma 3.3.1 *Let p be a point on an edge $e \in \Omega$ such that there is a vertex v of Ω visible from p and such that $d(p, v) \leq 15$. Then there is a point $q \in S^e$ within distance δ from p.*

Proof: We only need to show that every such point p lies on $\hat{e}$ because, by construction, for every point $q \in \hat{e}$, there is a point $q' \in S^e$ such that $d(q, q') \leq \delta$. Since $d(p, v) \leq 15$ and v is visible from p, the above algorithm would have been marked p, implying that $p \in \hat{e}$. □

Lemma 3.3.2 *Let Π be an optimal ε-robust path from I to F, and let $\langle X_1, \ldots, X_k \rangle$ be the sequence of positions on Π s.t. $\text{LOC}(X_i) \in \partial\Omega$ for every $1 \leq i \leq k$. There is a node $Y_i \in V$ such that*

(i) If $\text{LOC}(X_i)$ *is a vertex* v *of* Ω, *then* $\text{LOC}(Y_i) = v$ *and* $|\text{VEC}(Y_i) - \text{VEC}(X_i)| \leq \delta$; *or*

(ii) if $\text{LOC}(X_i)$ *is an interior point of an obstacle edge* e, *then* $\text{LOC}(Y_i) \in e$, $d(\text{LOC}(Y_i), \text{LOC}(X_i)) \leq \delta$, *and* $\text{VEC}(Y_i) = \text{VEC}(X_i)$.

Proof: If $\text{LOC}(X_i)$ is a vertex, the lemma follows from the definition of V_1. If $p = \text{LOC}(X_i)$ lies in the interior of an obstacle edge e, then p lies on a semi-free C-segment, a good anchored C-segment, or a bad anchored C-segment. In the last case, X_i is a node in V_2. In the first two cases, there is a vertex $v \in \Omega$ so that v is visible from p and that $d(p, v) \leq 15$. By Lemma 3.3.1 and the definition of V_3, there exists a node $Y_i \in V_3$ so that $d(\text{LOC}(Y_i), \text{LOC}(X_i)) \leq \delta$ and $\text{VEC}(Y_i) = \text{VEC}(X_i)$. This completes the proof of the lemma. □

Next, we prove that the size of V is $O(n/\delta)$. Since $|V_1| = O(n/\delta)$ and $|V_2| = O(n)$, it suffices to bound the size of V_3.

Lemma 3.3.3 $\sum_e |S^e| = O(n/\delta)$.

Proof: For each edge $e \in \Omega$, let $\hat{e}$ denote the marked portions of e, $\#\hat{e}$ the number of connected components of $\hat{e}$, and $\|\hat{e}\|$ the total length of $\hat{e}$. For each connected component γ of $\hat{e}$, the algorithm chooses $2 + \|\gamma\|/\delta$ points. Therefore, $|S^e| \leq 2\#\hat{e} + \|\hat{e}\|/\delta$.

The total measure of points marked in the first stage is at most $60n$. Recall that for each vertex v, $e \in E_v$ if $e \cap D_v$ contains at least one point whose distance to both endpoints of e is more than 30. Thus none of the endpoints

of E_v lie inside D_v, and the measure of points in $(\bigcup E_v) \cap D_v$ that are visible from v is at most 30π (the length of the perimeter of a disk of radius 15); see Figure 3.8. Therefore

$$\sum_{e \in \Omega} \|\hat{e}\| \leq 60n + 30\pi n = O(n).$$

Next, for each connected component $\gamma \in \hat{e}$, if γ is the first or the last connected component of $\hat{e}$, we charge γ to e itself. Otherwise, charge γ to any of the vertices that marked it in the second stage. We will prove in Lemma 3.3.4 below that each vertex is charged by at most 10 connected components, so

$$\sum_{e \in \Omega} \#\hat{e} = 2n + 10n = O(n).$$

This completes the proof. □

Lemma 3.3.4 *Each obstacle vertex is charged by at most* 10 *connected components.*

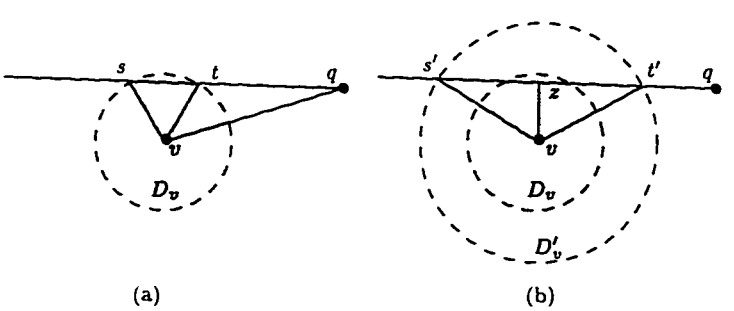

Figure 3.9: $d(s,t) < 15$ and $\angle s'vt' > \pi/2$.

Proof: Since the edges of E_v are disjoint and do not contain any of its endpoints inside D_v, v marks at most one connected component of each edge in E_v;

see Figure 3.8. That is, for each edge $e \in E_v$, either v marks the entire chord $e \cap D_v$ or it does not mark any point of e. Hence, the number of connected components charged to v is bounded by the number of edges in E_v that are visible from v.

Partition the set of edges visible from v into two subsets Γ_1 and Γ_2. An edge e is in Γ_1 if the length of the chord $e \cap D_v$ is at least 15, and in Γ_2 otherwise. Each edge $e \in \Gamma_1 \cup \Gamma_2$ splits the circle ∂D_v into two circular arcs; we refer to the shorter one as the *cap* induced by e. Since the edges in $\Gamma_1 \cup \Gamma_2$ are visible from v, the caps induced by them are pairwise disjoint.

Each cap induced by an edge of Γ_1 spans an angle of $\geq \pi/3$ because the length of the chord $e \cap D_v$ is at least 15 and the radius of D_v is 15. Since the caps are disjoint, $|\Gamma_1| \leq 6$.

Next, let e be an edge of Γ_2. Let p (resp. q) denote the left (resp. right) endpoint of e, and let s (resp. t) denote the left (resp. right) endpoint of $e \cap D_v$; see Figure 3.9a. By definition, $d(s,t) < 15$, and therefore $\angle stv, \angle tsv > \pi/3$. Since $e \in E_v$, e contains a point u such that $d(u,p), d(u,q) > 30$. Hence, $d(s,q) = d(s,u) + d(u,q) > 30$, and

$$\begin{aligned} d(v,q) &= \sqrt{d(s,q)^2 + d(v,s)^2 - 2\,d(s,q)\,d(v,s)\cos\angle vst} \\ &> \sqrt{30^2 + 15^2 - 2 \cdot 30 \cdot 15 \cdot \cos\pi/3} \\ &= 15\sqrt{3}. \end{aligned}$$

Similarly, one can show that $d(v,p) > 15\sqrt{3}$.

Draw a disk D'_v of radius $15\sqrt{3}$ centered at v. Since $d(v,p), d(v,q) > 15\sqrt{3}$, the endpoints of every edge in Γ_2 lie outside D'_v. Moreover, each edge of Γ_2 is

visible from v, the caps of $\partial D'_v$ induced by the edges of Γ_2 are also pairwise disjoint. For an edge e, let s', t' be the endpoints of $e \cap D'_v$, and z be the midpoint of the segment $s't'$; See Figure 3.9b. Since $d(v,z) \leq 15$ and $d(v,t') = 15\sqrt{3}$,

$$\angle s'vt' = 2\angle zvt' = 2 \cdot \cos^{-1} \frac{d(v,z)}{d(v,t')} \geq 2 \cdot \cos^{-1} \frac{1}{\sqrt{3}} > \pi/2.$$

Hence, the cap of $\partial D'_v$ induced by $e \in \Gamma_2$ spans an angle of $> \pi/2$, which implies $|\Gamma_2| \leq 4$. This completes the proof of the lemma. □

We now show that the node set V_3 can be computed in time $O(n^2 \log n + n/\delta)$. For every vertex v, the set E_v can be computed in $O(n)$ time in a straight-forward manner. Let $E'_v = \{e \cap D_v \mid e \in E_v\}$. The portions to be marked are the subset of edges of E'_v that are visible from v with respect to the edge set E'_v. (Recall that for every edge $e \in E'_v$, either every point on e is visible from v, or no point on e is visible from v.) This set can be computed in $O(n \log n)$ time by performing an angular sweep around v. Let $\rho(\theta)$ be the ray emanating from v in direction θ. Let $\theta_1, \ldots, \theta_k$ be the orientations such that $\rho(\theta)$ passes through an endpoint of an edge in E'_v. We sweep the plane with the ray $\rho(\theta)$ by varying θ from 0 to 2π. For each θ, we maintain the edges of E'_v intersecting $\rho(\theta)$, sorted in the order they intersect $\rho(\theta)$. Since the segments of E'_v are pairwise disjoint, the ordering changes only at θ_i's. Let $e_i \in E'_v$ be the first edge in this ordering in the interval $[\theta_i, \theta_{i+1})$. We mark e_i. At each θ_i, we can update the ordering in time $O(\log n)$. Repeating this process for all the vertices of Ω, V_3 can be computed in time $O(n^2 \log n + n/\delta)$. Hence, we conclude the following

Lemma 3.3.5 *The node set V can be computed in time $O(n^2 \log n + n/\delta)$.*

3.3.2 Computing the edge set

We now describe how to compute the edge set E. For each pair of positions $X, Y \in V$, we compute all $O(1)$ Dubins paths from X to Y, check which of them are feasible, and select the one with the minimum arc length. The only nontrivial step is to determine whether a given Dubins path is feasible. We will consider CCC and CLC paths separately.

Testing CCC paths. CCC paths can be further classified into two subcategories, as follows. If we label a clockwise (resp. counterclockwise) oriented C-segment as C^+ (resp. C^-), then a CCC path is either $C^+C^-C^+$ type or $C^-C^+C^-$ type. We consider only $C^+C^-C^+$ paths; $C^-C^+C^-$ paths can be handled in a similar manner.

For each position X, let C_X denote the clockwise oriented circle passing through X, and let ϕ_X^+ (resp. ϕ_X^-) be the intersection point of C_X and $\partial\Omega$ immediately after (resp. before) $\textsc{loc}(X)$, so the interiors of the arcs $C_X[\textsc{loc}(X), \phi_X^+]$ and $C_X[\phi_X^-, \textsc{loc}(X)]$ do not intersect $\partial\Omega$ (see Figure 3.10a); ϕ_X^+ and ϕ_X^- can be computed in $O(n)$ time.

Let $w_1w_2w_3$ be a $C^+C^-C^+$ path from a position X to another position Y, with a_i being the common endpoint of w_i and w_{i+1}, for $i = 1, 2$. Obviously w_1 (resp. w_3) does not intersect Ω if and only if $a_1 \in C_X[\textsc{loc}(X), \phi_X^+]$ (resp. $a_2 \in C_Y[\phi_Y^-, \textsc{loc}(Y)]$); see Figure 3.10a. After computing ϕ_X^+ and ϕ_X^- for all $O(n/\delta)$ positions in time $O(n^2/\delta)$, given a $C^+C^-C^+$ path, we can check in

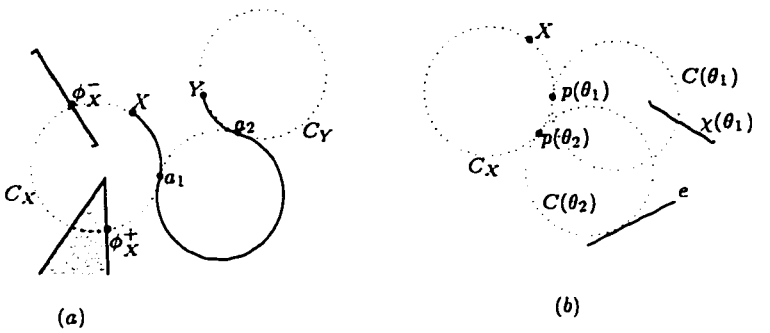

Figure 3.10: (a) $C^+C^-C^+$ path; (b) $p(\theta), C(\theta)$, and a critical orientation θ_2.

$O(1)$ time whether its first and last C-segments intersect Ω. Next, we describe how to test whether the middle C-segment of a $C^+C^-C^+$ path intersects Ω.

Fix a position X. We construct a linear-size data structure, in $O(n \log n)$ time, which, given a position Y, can determine in $O(\log n)$ time whether the middle C-segment of the $C^+C^-C^+$ path from X to Y intersects any obstacle edge.

For an orientation θ, $0 \leq \theta < 2\pi$, let $p(\theta)$ be the point on C_X such that the arc length of $C_X[\text{LOC}(X), p(\theta)]$ is θ. Let $C(\theta)$ be the counterclockwise-directed circle tangent to C_X at $p(\theta)$. Set $\chi(\theta)$ to be the first edge of Ω intersected by $C(\theta)$, as one walks along it (in the counterclockwise direction) starting from $p(\theta)$. If there is no such edge, then $\chi(\theta)$ is undefined (see Figure 3.10b).

We call an orientation θ *critical* if $C(\theta)$ is either tangent to an obstacle edge or it passes through an obstacle vertex. Let $\langle \theta_1, \theta_2, \ldots, \theta_k \rangle$ be the sequence of critical angles sorted in the increasing order. Using a simple continuity argument, we can prove the following.

Lemma 3.3.6 *For all orientations θ in any interval $[\theta_i, \theta_{i+1})$, the set of ob-*

stacle edges that intersect $C(\theta)$, and the order in which they intersect $C(\theta)$, remains the same.

This lemma implies that the value of $\chi(\theta)$ remains the same for all orientations within each interval $[\theta_i, \theta_{i+1})$.

Lemma 3.3.7 *There are $O(n)$ critical orientations.*

Proof: For an edge $e \in \Omega$, let D_e be the Minkowski sum of e and the unit-radius disk. Let C'_X be the circle concentric with C_X and of radius 2. $C(\theta)$ is tangent to e, or passes through an endpoint of e, if and only if $C(\theta)$ is centered at an intersection point of C'_X and ∂D_e. There are at most 4 intersection points between C'_X and ∂D_e. Thus an edge e can contribute at most $O(1)$ critical orientations, resulting in $O(n)$ critical orientations for all edges. □

These $O(n)$ critical orientations can be computed in $O(n)$ time. By sweeping the circle $C(\theta)$ for $\theta \in [0, 2\pi)$, we can compute the values of χ for all $O(n)$ intervals $[\theta_i, \theta_{i+1})$ in $O(n \log n)$ time, as follows. For each θ, we maintain the set of edges intersecting $C(\theta)$, sorted in the order in which they intersect $C(\theta)$. By Lemma 3.3.6, this ordering changes only at critical orientations. At each critical orientation, we can update the ordering in $O(\log n)$ time, thus, spending a total of $O(n \log n)$ time. We record the values of χ for each interval $[\theta_i, \theta_{i+1})$.

Now, given a $C^+C^-C^+$ Dubins path $w_1w_2w_3$, we first compute the orientation θ of a_1, the common endpoint of w_1 and w_2. Using the above data structure, we can determine $e = \chi(\theta)$ in $O(\log n)$ time by a binary search.

Finally, we check in $O(1)$ time whether w_2 intersects the edge e (That is, we check whether a_2, the common endpoint of w_2 and w_3, lies before the intersection point of $C(\theta)$ and e, in which case w_2 does not intersect any edge of Ω). This completes the description of the data structure.

We can thus conclude that for all $X, Y \in V$, we can determine in time $O((n^2/\delta^2)\log n)$ whether there is a feasible CCC-path from X to Y.

Testing CLC paths. There are four types of CLC paths, namely C^+LC^+, C^+LC^-, C^-LC^+ and C^-LC^-. Consider C^+LC^+ paths. Let $w_1w_2w_3$ be a C^+LC^+ path. After $O(n^2/\delta)$ preprocessing as above, we can easily determine whether w_1 or w_3 intersects Ω. As for w_2, we construct a similar data structure. Fix a position X. For a given θ, we now define $\ell(\theta)$ to be the ray tangent to C_X and emanating from $p(\theta)$, and define $\chi(\theta)$ to be the first edge of Ω intersected by $\ell(\theta)$. An orientation θ is *critical* if $\ell(\theta)$ passes through a vertex of Ω. We can again construct a linear-size data structure in $O(n\log n)$ time that given a position Y, can determine in $O(\log n)$ time whether the line segment of C^+LC^+ path from X to Y intersects Ω.

Hence, we can compute in $O((n^2/\delta^2)\log n)$ time all pairs $X, Y \in V$ that admit a feasible C^+LC^+ path from X to Y. Repeating this procedure for other types of CLC paths, we can compute in $O((n^2/\delta^2)\log n)$ time all the pairs $X, Y \in V$ for which there is a feasible CLC path from X to Y.

After having computed the vertices and edges of G, we can compute a shortest path in G, using any standard shortest-path algorithm [26]. Putting everything together, we obtain the following result.

Theorem 3.3.8 *The graph G, as described above, can be constructed in time $O((n^2/\delta^2)\log n)$, and a shortest path from I to F in G can be computed in an additional $O(n^2/\delta^2)$ time.*

3.4 Error Analysis

In this section we prove that if δ is chosen to be at most $c\varepsilon^2$, where c is a sufficiently small constant independent of ε, then the above algorithm computes an $(1+\varepsilon)$-approximate shortest path from I to F. We first bound the change in the length of a Dubins path as we perturb its end-positions, and then we bound the length of the path computed by the above algorithm.

3.4.1 Error induced by a single Dubins path

To give an error bound for our approximation algorithm, we need to answer the following question: given two Dubins paths of the same type whose end-positions differ by a small amount, what is the difference in length of these two paths? Let X and X' be two positions. If $\text{LOC}(X') = \text{LOC}(X)$, we define $\Delta(X', X)$ to be $|\text{VEC}(X') - \text{VEC}(X)|$; and if $\text{VEC}(X') = \text{VEC}(X)$, then we define $\Delta(X', X)$ to be $\|\text{LOC}(X') - \text{LOC}(X)\|$. For two paths Π and Π', let $\Delta(\Pi', \Pi)$ be the difference in length of these two paths, i.e., $\Delta(\Pi', \Pi) = \Big|\|\Pi'\| - \|\Pi\|\Big|$. Two Dubins paths Π_{XY} and $\Pi_{X'Y'}$ of the same type from X to Y and from X' to Y', respectively, will be called *homotopic* if Π_{XY} can be continuously deformed to $\Pi_{X'Y'}$ in such a way that every intermediate path is also a Dubins path of the same type.

Lemma 3.4.1 *Let Π be a CLC path from a position I to a position F. Let I' and F' be positions such that $\Delta(I', I) = \delta_I$ and $\Delta(F', F) = \delta_F$, for any reals $\delta_I, \delta_F \geq 0$. Let Π' be the path from I' to F' of the same type as Π and homotopic to Π. Then*

$$\Delta(\Pi', \Pi) = O(\delta_I + \delta_F).$$

Proof: We will prove that if $\delta_F = 0$ (i.e., $F' = F$), then $\Delta(\Pi', \Pi) = O(\delta_I)$. By reversing the direction of Π, this also implies that $\Delta(\Pi', \Pi) = O(\delta_F)$ if $\delta_I = 0$. If both $\delta_I, \delta_F > 0$, then let Π'' be the CLC path from I' to F of the same type as Π. Then

$$\Delta(\Pi', \Pi) \leq \Delta(\Pi'', \Pi) + \Delta(\Pi', \Pi'') = O(\delta_I + \delta_F),$$

as claimed. We now assume $F' = F$, and set $\delta = \delta_I$.

We will prove the claim for the case in which I is located at an obstacle vertex u, i.e., $I = (u, \theta_I)$ and $|\text{VEC}(I') - \text{VEC}(I)| = \delta$. Let C_1 (resp. C_2) be the unit-radius circle containing the initial (resp. final) C-segment of Π, and let o_i (for $i = 1, 2$) denote the centers of C_i. Let C_1' be the circle containing the initial C-segment of Π', and let o_1' be the center of C_1'. If Π is a C^+LC^+ or C^-LC^- type path, then

$$\|\Pi\| = |\text{VEC}(I) - \text{VEC}(F)| + d(o_1, o_2),$$

in which case,

$$\Delta(\Pi, \Pi) \leq |\text{VEC}(I') - \text{VEC}(I)| + |d(o_1, o_2) - d(o_1', o_2)| \leq \delta + d(o_1, o_1').$$

By applying the cosine law to $\triangle o_1 u o_1'$ (see Figure 3.11(b)),

$$d(o_1, o_1') = \sqrt{1 + 1 - 2\cos\delta} = 2\sin(\delta/2) \leq \delta. \tag{3.1}$$

Therefore, $\Delta(\Pi', \Pi) \leq 2\delta$.

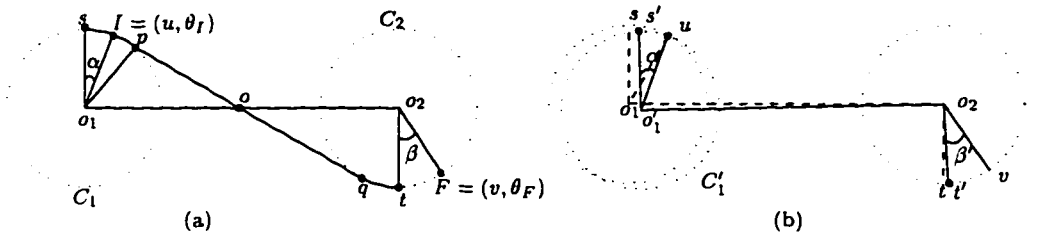

Figure 3.11: Bounding the difference in path length for CLC paths.

In the following we prove the lemma for the case when Π is a C^+LC^- type path; the other case, when Π is a C^-LC^+ type path, is symmetric. Let p (resp. q) be the initial (resp. final) point of the L-segment of Π, i.e., it is the point at which the common tangent of C_1^+ and C_2^- touches C_1 (resp. C_2). Let s be the point on C_1 such that $so_1 \perp o_1o_2$ and $\angle so_1p < \pi/2$, and let S be the position on C_1 corresponding to s, i.e., $\text{LOC}(S) = s$ and $\text{VEC}(S)$ is the orientation of the tangent to C_1 (assuming that C_1 is clockwise oriented) at s. We define a similar point t and position T on C_2; see Figure 3.11(a). Instead of considering Π, we consider the C^+LC^- path from S to T; and let ρ be the length of this path. If we measure the angles in counterclockwise direction and choose their values between $-\pi$ and π, then

$$\|\Pi\| = \rho + \alpha + \beta$$

(see Figure 3.11(a)), where $\alpha = \angle so_1u$ and $\beta = \angle to_2v$. We define ρ', α' and β' corresponding to Π'. Then $\|\Pi'\| = \rho' + \alpha' + \beta'$, and

$$\Delta(\Pi', \Pi) \leq |\rho' - \rho| + |\alpha' - \alpha| + |\beta' - \beta|. \tag{3.2}$$

We first bound $|\alpha' - \alpha|$. Applying the sine law to $\triangle o_1 o_2 o_1'$,

$$\frac{\sin \angle o_1 o_2 o_1'}{d(o_1, o_1')} = \frac{\sin \angle o_1' o_1 o_2}{d(o_1, o_2)}.$$

Using (3.1), the fact that $d(o_1, o_2) \geq 2$, and the inequality $\sin^{-1} x \leq 2x$ for any $0 \leq x \leq 1$, we obtain

$$\angle o_1 o_2 o_1' \leq \sin^{-1} \frac{\delta}{2} \leq \delta.$$

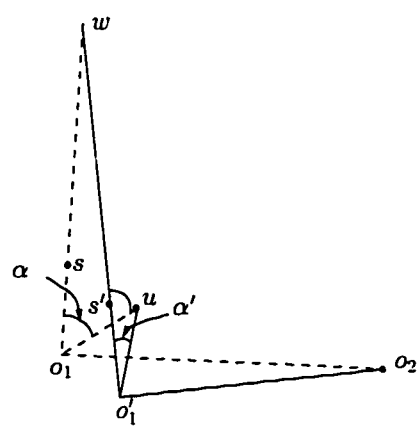

Figure 3.12: Bounding $|\alpha' - \alpha|$.

Let w be the intersection point of lines supporting the segments so_1 and $s'o_1'$. Since $so_1 \perp o_1 o_2$ and $s'o_1' \perp o_1' o_2$, $\angle o_1 w o_1' = \angle o_1 o_2 o_1' \leq \delta$; see Figure 3.12. Moreover, $\alpha' + \angle o_1 u o_1' = \alpha + \angle o_1 w o_1'$, therefore

$$|\alpha' - \alpha| \leq \angle o_1 u o_1' + \angle o_1 w o_1' = O(\delta). \tag{3.3}$$

Similarly, we can show that $|\beta' - \beta| = O(\delta)$.

Next, we bound $|\rho' - \rho|$. Notice that

$$\rho = 2\left(d(o, p) + \|C_1[s, p]\|\right).$$

Let $\mu = d(o_1, o) = d(o_1, o_2)/2$, then $\angle so_1p = \angle poo_1 = \sin^{-1}(1/\mu)$, and we have

$$\rho = 2\left(\sin^{-1}\left(\frac{1}{\mu}\right) + \sqrt{\mu^2 - 1}\right).$$

Similarly, if we let $\nu = d(o_1', o_2)/2$, then

$$\rho' = 2\left(\sin^{-1}\left(\frac{1}{\nu}\right) + \sqrt{\nu^2 - 1}\right).$$

As in (3.1), $d(o_1', o_1) \leq \delta$, which implies that $\nu - \mu \leq \delta/2 \leq \delta$. Consider the function

$$f(x) = \sin^{-1}\left(\frac{1}{x}\right) + \sqrt{x^2 - 1}.$$

Then $|\rho' - \rho| = 2|f(\nu) - f(\mu)|$. Since both $f(x)$ and it derivative are monotonically increasing,

$$|\rho' - \rho| \leq 2(f(\mu + \delta) - f(\mu))$$

and $|\rho' - \rho|$ maximizes as μ tends to infinity. Using Taylor expansion, one can write

$$f(x) = x + \sum_{i=1}^{\infty} \frac{c_i}{x^{2i-1}},$$

where $|c_i|$'s are monotonically decreasing with i. Thus

(3.4)

$$|\rho' - \rho| \leq \lim_{\mu \to \infty} 2\left(\mu + \delta + \sum_{i=1}^{\infty} \frac{c_i}{(\mu + \delta)^{2i-1}} - \left(\mu + \sum_{i=1}^{\infty} \frac{c_i}{\mu^{2i-1}}\right)\right) = O(\delta).$$

Combining together (3.3) and (3.4), we obtain that

$$\Delta(\Pi', \Pi) = O(\delta).$$

This completes the proof of the lemma. □

Lemma 3.4.2 *Let Π be a CCC path from a position I to a position F. Let I' and F' be positions such that $\Delta(I', I) = \delta_I$ and $\Delta(F', F) = \delta_F$, for any reals $\delta_I, \delta_F \geq 0$. Let Π' be the Dubins path from I' to F' of the same type as Π and homotopic to Π. Then*

$$\Delta(\Pi', \Pi) = O(\sqrt{\delta_I} + \sqrt{\delta_F}).$$

Proof: As in the proof of Lemma 3.4.1, we only need to prove the lemma for the case when $\delta_I > 0$ and $\delta_F = 0$. Let $\delta = \delta_I$. We assume that I is located at an obstacle vertex, thus $\text{LOC}(I') = \text{LOC}(I)$ and $|\text{VEC}(I') - \text{VEC}(I)| = \delta$. We prove the lemma for the case when Π is a $C^-C^+C^-$ path; the other case, when Π is a $C^+C^-C^+$ path, is symmetric.

Let p (resp. q) be the initial (resp. final) location of the path Π. Let o_i be the center of the unit circle containing the ith C-segment of Π. Consider the triangle $\Delta o_1 o_2 o_3$; see Figure 3.13(a). Without loss of generality, assume that p lies inside $\Delta o_1 o_2 o_3$ and q does not; other cases can be handled similarly. Let b_i be the angle of the triangle adjacent to o_i.

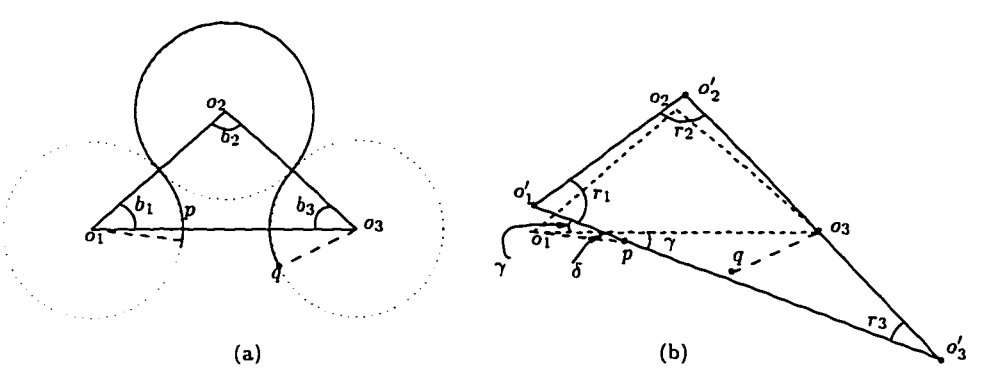

Figure 3.13: Bounding the difference in path length for CCC paths.

Then

$$\begin{aligned}
\|\Pi\| &= b_1 + \angle po_1o_3 + 2\pi - b_2 + b_3 + \angle o_1o_3q \\
&= (b_1 + b_3) + 2\pi - b_2 + \angle po_1o_3 + \angle o_1o_3q \\
&= \pi - b_2 + 2\pi - b_2 + \angle po_1o_3 + \angle o_1o_3q \\
&= 3\pi - 2b_2 + \angle po_1o_3 + \angle o_1o_3q.
\end{aligned}$$

Let o_1', o_2' and o_3 be the centers of circles containing the C-segments of Π'. Notice that $\angle o_1'po_1 = \delta$; see Figure 3.13(b). Consider the triangle $\Delta o_1'o_2'o_3'$, where o_3' is the intersection point of the line supporting the segment $o_1'p$ and the line supporting the segment $o_2'o_3$; assume that o_1' is situated so that o_3 lies between o_2 and o_3'. Let r_i be the angle of the triangle adjacent to o_i'. Then

$$\begin{aligned}
\|\Pi'\| &= r_1 + 2\pi - r_2 + \angle o_2'o_3o_1 + \angle o_1o_3q \\
&= r_1 + 2\pi - r_2 + r_3 + \gamma + \angle o_1o_3q \\
&= r_1 + 2\pi - r_2 + r_3 + \angle o_1'po_1 + \angle po_1o_3 + \angle o_1o_3q \\
&= (r_1 + r_3) + 2\pi - r_2 + \delta + \angle po_1o_3 + \angle o_1o_3q \\
&= 3\pi - 2r_2 + \delta + \angle po_1o_3 + \angle o_1o_3q.
\end{aligned}$$

Thus

$$\Delta(\Pi', \Pi) = |\delta + 2b_2 - 2r_2| \leq \delta + 2|b_2 - r_2|.$$

Let $\rho = d(o_1, o_3)$ and $\rho' = d(o_1', o_3)$. Applying the cosine law to $\Delta o_1o_2o_3$ and $\Delta o_1'o_2'o_3$, and using the fact that $d(o_1, o_2) = d(o_2, o_3) = d(o_1', o_2') = d(o_2', o_3) =$

2, we obtain

$$b_2 = \cos^{-1}\left(\frac{2^2+2^2-\rho^2}{2\cdot 2\cdot 2}\right) = \cos^{-1}\left(1-\frac{\rho^2}{8}\right), \text{and}$$

$$r_2 = \cos^{-1}\left(\frac{2^2+2^2-\rho'^2}{2\cdot 2\cdot 2}\right) = \cos^{-1}\left(1-\frac{\rho'^2}{8}\right).$$

Define a function

$$f(x) = \cos^{-1}\left(1-\frac{x^2}{8}\right).$$

Then $|b_2-r_2| = |f(\rho)-f(\rho')|$. Since $f(x)$ is monotonically increasing, $|b_2-r_2| \leq f(\rho)-f(\rho-\delta)$. Moreover, $\frac{df}{dx} = 8 \,/\, \sqrt{1-\frac{x^2}{16}}$ is also monotonically increasing and $\rho \leq 4$, therefore $|b_2 - r_2|$ maximizes at $\rho = 4$. Thus

$$\begin{aligned}
|b_2 - r_2| &\leq \cos^{-1}\left(1-\frac{4^2}{8}\right) - \cos^{-1}\left(1-\frac{(4-\delta)^2}{8}\right) \\
&\leq \pi - \cos^{-1}(-1+\delta) \\
&= \pi - \left(\pi - 2\sqrt{\sin^{-1}\frac{\delta}{2}}\right) \\
&= 2\sqrt{\sin^{-1}\frac{\delta}{2}} \quad \leq 2\sqrt{\delta},
\end{aligned}$$

as desired. □

Remark 3.4.3 Notice that $\Delta(\Pi,\Pi') = O(\sqrt{\delta})$ only if the distance between the centers o_1 and o_3 is almost 4. If $d(o_1,o_3) \leq 4-c$ for some constant $c > 0$, then $\Delta(\Pi,\Pi') \approx \delta/\sqrt{c}$.

3.4.2 Goodness of our approximation

Lemma 3.4.4 *If $\delta = c\varepsilon^2$, where c is a sufficiently small constant , then there exists a path from I to F in the graph G computed in Section 3.3 whose length is at most $(1+\varepsilon)$ times the length of an optimal ε-robust path from I to F.*

Proof: Let $\Pi = \Pi_1 \| \ldots \| \Pi_k$ be an optimal ε-robust path from I to F, where each Π_i is a Dubins path from a position X_{i-1} to a position X_i, such that $X_0 = I$, $X_k = F$ and $\text{LOC}(X_i) \in \partial\Omega$, for $0 < i < k$. By Lemma 3.3.2, there exist graph nodes $Y_0 = I, Y_1, \ldots, Y_{k-1}, Y_k = F$, such that $\Delta(Y_i, X_i) \leq \delta$, such that if $\text{LOC}(X_i)$ is a vertex of Ω then $\text{LOC}(Y_i) = \text{LOC}(X_i)$, and such that if $\text{LOC}(X_i)$ is an interior point of an edge of Ω then $\text{VEC}(Y_i) = \text{VEC}(X_i)$. Since each Π_i is an ε-robust path and we are going to choose $\delta = c\varepsilon^2 < \varepsilon$, there is a feasible Dubins path Π_i' of the same type as Π_i from Y_i to Y_{i+1}, for $0 \leq i \leq k-1$. Therefore (Y_i, Y_{i+1}) is an edge in G, and $\Pi' = \Pi_1' \| \ldots \| \Pi_k'$ is the desired path from I to F in G. To prove the lemma, it suffices to show that $\|\Pi_i'\| \leq (1+\varepsilon)\|\Pi_i\|$, for $1 \leq i \leq k$, provided we choose $\delta = c\varepsilon^2$ small enough.

If Π_i is a CLC path, by Lemma 3.4.1, $\Delta(\Pi_i', \Pi_i) \leq O(\delta)$. Since Π_i is an ε-robust path, its length is at least ε. Therefore,

$$\|\Pi_i'\| \leq \|\Pi_i\| + O(\delta) \leq \|\Pi_i\| + \varepsilon^2 \leq (1+\varepsilon)\|\Pi_i\|,$$

provided the constant c is chosen sufficiently small. If Π_i is a CCC path, by Lemma 3.4.2, $\Delta(\Pi_i', \Pi_i) \leq O(\sqrt{\delta})$. But the length of a CCC path is at least π, therefore

$$\|\Pi_i'\| \leq \|\Pi_i\| + O(\sqrt{\delta}) \leq \|\Pi_i\| + \varepsilon \leq \left(1 + \frac{\varepsilon}{\pi}\right)\|\Pi_i\| \leq (1+\varepsilon)\|\Pi_i\|,$$

provided c is chosen small enough. This completes the proof of the lemma. □

Plugging $\delta = O(\varepsilon^2)$ in Theorem 3.3.8, we obtain the following result.

Theorem 3.4.5 *Given a polygonal obstacle environment Ω, an initial position I, a final position F, and a parameter ε, we can compute in time $O((n^2/\varepsilon^4)\log n)$ a feasible path from I to F whose arc length is at most $(1+\varepsilon)$ times the length of an optimal ε-robust path.*

Remark 3.4.6 Recall that the running time of the algorithm is $O((n^2/\varepsilon^4)\log n)$ because we choose $\delta = \varepsilon^2$, and the graph G has $O((n/\delta))$ vertices and in the worst-case every pair of vertices is connected by an edge. If the distance between the centers of initial and final circles of CCC-type paths for most pairs of vertices is not close to 4, one can show that it suffices to add edges between $O((n^2/\delta)\log(1/\delta))$ pairs of vertices, and that these pairs can be computed in time $O((n^2/\delta^2)\log(1/\delta))$. In this case the time complexity improves to $O((n^2/\varepsilon^2)\log n \log(1/\varepsilon))$.

3.5 Computing Near Optimal Robust Paths

The path computed by the above algorithm is not necessarily robust because some of the edges in G may not correspond to robust paths. We can compute a graph $G' = (V, E')$, where E' is the set of edges corresponding to $(\varepsilon/2)$-robust paths. An easy argument shows that if δ is chosen correctly, there is an $(\varepsilon/2)$-robust path in G whose length is at most $(1+\varepsilon)$ times the length of an optimal ε-robust path from I to F.

Next we show that E' can be computed in $O((n^{2.5}/\varepsilon^4)\log n)$ time. For each pair of positions $X, Y \in V$, we compute all $O(1)$ Dubins paths from X to Y, check which of them are $(\varepsilon/2)$-robust, and select the one with the minimum arc length. The only nontrivial step is to determine whether a given Dubins path is $(\varepsilon/2)$-robust.

Let Π_{XY} be a Dubins path from X to Y. We can assume that $\text{LOC}(X)$ and $\text{LOC}(Y)$ are obstacle vertices. The other cases when $\text{LOC}(X)$ or $\text{LOC}(Y)$ lie in the interior of obstacle edges can be dealt in a similar manner. Let γ_{XY} be the region formed by the set of Dubins paths $\Pi_{X'Y'}$ such that $\Delta(X, X'), \Delta(Y, Y') \leq \varepsilon/4$ and $\Pi_{X'Y'}$ can be obtained by deforming Π_{XY} continuously. The boundary of γ_{XY} consists of $O(1)$ x-monotone algebraic arcs, each of $O(1)$ degree, and they can be computed in $O(1)$ time.[1] Π_{XY} is $(\varepsilon/2)$-robust if and only if γ_{XY} does not intersect the interior of Ω. Since $\text{LOC}(X)$ and $\text{LOC}(Y)$ lie on the boundary of obstacles, γ_{XY} intersects the interior of any obstacle if and only if any of the obstacle edges intersect the interior of γ_{XY}. We thus have the following intersection-detection problem at hand: Let $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ be a set of m regions, each of whose boundary consists of $O(1)$ algebraic arcs of constant degrees, and let S be a set of n disjoint line segments in the plane. Report all regions in Γ whose interiors do not intersect any segment of S. We present an $O((m\sqrt{n} + n)\log n)$-time algorithm to report such a subset. Since $m = O(n^2/\varepsilon^4)$ in our case, we conclude that we can computer all $(\varepsilon/2)$-robust paths in time $O((n^{2.5}/\varepsilon^4)\log n)$.

[1] Note that there exist X' and Y' such that no $\Pi_{X'Y'}$ can be obtained by a continuous deformation of Π_{XY}. In this case, we consider Π_{XY} non-robust even if there is a free Dubins path from X' to Y'. These cases can be detected in $O(1)$ time.

We now describe an algorithm for the intersection-detection problem just described. It suffices to describe an $O(n \log n)$-time algorithm for the case when $m = \sqrt{n}$, for otherwise we can partition Γ into $\lceil m/\sqrt{n} \rceil$ subsets, $\Gamma_1, \ldots, \Gamma_s$, each of size at most $\sqrt{n}$, and solve the intersection-detection problem for each Γ_i and S separately. The total running time is obviously $O((m\sqrt{n} + n) \log n)$.

Let E be the set of x-monotone arcs bounding the regions in Γ. A segment $e \in S$ intersects the interior of a region $\gamma \in \Gamma$ if at least one of the following two conditions is satisfied: (i) An endpoint of e lies in the interior of γ, or (ii) e intersects the boundary of γ. It is possible to check both of these conditions for all regions in Γ in $O(n \log n)$ time, using a single sweep-line algorithm. But for the sake of clarity, we explain how to check each of the two conditions separately. The first condition can be checked in $O(n \log n)$ time by a variant of the batched point-location algorithm by Preparata [83]. It basically sweeps a vertical line from left to right and maintains the subset of regions that intersect the sweep-line. Whenever the sweep-line encounters an endpoint p of S, it reports and deletes all the regions of Γ whose interior contains p. These steps can be implemented efficiently using interval trees or segment trees. We omit the rather easy and standard details from here. The total running time of the algorithm is $O((m^2 + n + k) \log(m + n))$, where k is the number of regions in Γ that contain an endpoint of S. Since $m = \sqrt{n}$, and each region of Γ is a semi-algebraic region of constant description complexity, $k = O(m) = O(\sqrt{n})$. Hence, the total time spent is $O(n \log n)$.

Next, we explain how to detect condition (ii). This can be done by modifying the Bentley-Ottman [9] algorithm for segment-intersection reporting, as

follows. We sweep a vertical line from left to right and store the the arcs of $E \cup S$ intersecting the sweep-line in a height balanced tree T, sorted in y-direction. The algorithm maintains the invariant that none of the arcs of E intersecting the sweep-line intersects any segment of S. A region γ is deleted from Γ as soon as we detect an intersection between $\partial\gamma$ and S; all the boundary arcs of γ are deleted from E as well. The sweep-line stops at the endpoints of $E \cup S$ and the intersection points of arcs in E. At the left (resp. right) endpoint of an arc $e \in E$, we insert e into T (resp. delete e from T). We do the same at the endpoints of S. At an intersection point σ of two arcs $\rho_1, \rho_2 \in E$, we swap the order of ρ_1 and ρ_2 in T. Whenever one of the adjacent element of an active arc $\rho \in E$ changes (because of insertion, deletion, or swapping of two arcs), we check whether the new adjacent element is a segment $e \in S$. If e and ρ intersect, we delete ρ from E and T. Let γ be the region bounded by ρ. We report and delete γ from Γ, and delete all the edges bounding γ. Note that deletion of these arcs from T may change the adjacent elements of other arcs stored in T, so we have to check for their intersections, but this time can be charged to the arcs deleted. Since each arc of Γ is deleted only once, and there are $m = \sqrt{n}$ arcs, the total time spent in this step is $O(\sqrt{n}\log(m+n))$. The sweep-line stops in at most $O(m^2 + n)$ points, hence the overall time spent is also $O(n \log n)$. Putting all the steps together, we obtain

Theorem 3.5.1 *Given a polygonal obstacle environment Ω, an initial position I, a final position F, and a parameter ε, we can compute in time $O((n^{2.5}/\varepsilon^4)\log n)$ a feasible $(\varepsilon/2)$-robust path from I to F whose arc length is at most $(1+\varepsilon)$ times the length of an optimal ε-robust path.*

Chapter 4

On-Line Navigation Through Regions of Variable Densities

Let B be a point robot moving in the plane. A polygon of *density* > 1 is called an *obstacle*, while the free space has a density of 1. An obstacle may contain other obstacles within itself, but these obstacles must have higher densities (see Figure 4.1; heavier shades represent higher densities). An obstacle is *impenetrable* if its density is ∞. The *effort* of traveling a Euclidean distance of d in a region of density ρ is defined to be ρd.

The robot does not know the positions, the extents, or the densities of these obstacles beforehand; rather, it finds out about obstacles as it encounters them. We assume that *when the robot first touches an obstacle, it is given the shape, the size, the position and the density of the obstacle.*

A *scene* S consists of a start point s, a target t, and a set of obstacles. Let $e_A(S)$ be the total effort spent by the robot traveling from s to t in scene S, using strategy A, and let $e_O(S)$ be the minimum effort. In this case, the

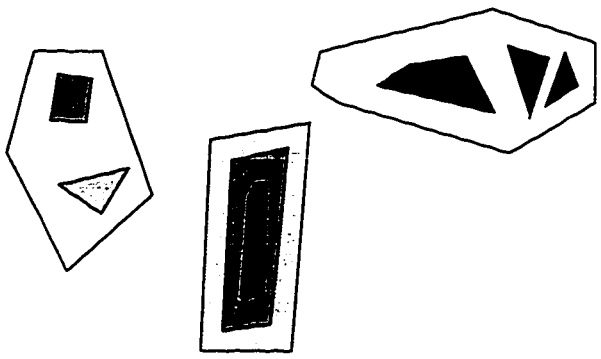

Figure 4.1: The obstacles are of variable densities and may contain obstacles of higher densities.

competitive ratio is defined to be

$$\max_{S \in \mathcal{S}(n)} \frac{e_A(S)}{e_O(S)},$$

where $\mathcal{S}(n)$ denote the set of scenes in which the Euclidean distance between s and t is n.

The *wall problem with penetrable obstacles* (WPPO) extends the original *wall problem* [11] to deal with penetrable obstacles of various densities. The objective is to minimize the effort instead of the Euclidean distance. As in the wall problem, the obstacles are disjoint oriented rectangles and the target is an infinite oriented line. In WPPO, obstacles do not contain other obstacles within. For this problem, we give an optimal on-line algorithm (see Section 4.1) whose competitive ratio is $\Theta(\sqrt{n})$, by modifying the *sweeping strategy* of [11]. We further generalize WPPO to the *recursive wall problem* where obstacles may contain other obstacles of higher densities (see Section 4.2). We give a lower bound on the competitive ratio for this problem.

4.1 The Wall Problem with Penetrable Obstacles

For convenience, we put the scene in an xy-coordinate system, and let the start point s be at $(0, 0)$. The target t is an infinite line, parallel to the y-axis and at distance n from s. The obstacles are oriented rectangles and are disjoint (This implies that when two obstacles touch, the robot can "squeeze" through them). Each obstacle is associated with a density, and does not contain other obstacles. Following the convention of [11], let the *width* of an obstacle be its dimension parallel to the x axis, and the *height* of an obstacle its dimension parallel to the y axis. For convenience, we use "go north" (resp. "go south") when the robot goes along (resp. against) y axis. We use "east" to indicate the direction of x axis.

We also assume that *the width and the height of every obstacle is at least one.*

4.1.1 The generalized sweeping strategy

We generalize the original sweeping strategy of [11] to adapt to WPPO. We also maintain four variables, a *window size*, W, a *threshold*, τ, a *sweep direction*, and a *sweep counter*. A sweeping window of size W is defined by the two window sides $y = \pm W/2$. Initially, W is set to n, the sweep direction is south, and the sweep counter is set to 0. The threshold τ is always set to $W/\sqrt{n}$. Whenever the sweep counter reaches $\sqrt{n}$, it is reset to 0 and the window size W and the threshold τ get doubled.

Starting from point s, the robot travels due east until it either reaches t or hits an obstacle, say at point (x, y). Let y_n and y_s be the y-coordinates of the north and south corners of the obstacle. Let w denote the width of the obstacle. Let E be the effort of going through the obstacle, arriving at $(x+w, y)$, and let N (resp. S) be the effort of traveling along the height of the obstacle, reaching the point (x, y_n) (resp. (x, y_s)). When the robot encounters an obstacle, the following three rules are applied.

Rule 1: If $\min(E, N, S) < \tau$, do the following. If E is the smallest of the three, the robot goes through the obstacle, arriving at $(x+w, y)$. Otherwise it travels along the boundary of the obstacle through the nearest corner, and arrives at $(x + w, y)$. It continues to travel due east until it hits another obstacle. (See Figure 4.2(a).)

Rule 2: If $y_n > W/2$, $y_s < -W/2$, and $E > W$, do the following. Without loss of generality, we can assume that $|y_s| < y_n$. If $E < S$, the robot goes through the obstacle, arriving at $(x + w, y)$. Otherwise, it travels along the height of the obstacle to the south corner, and then along the width of the obstacle to $(x+w, y_s)$. In the first case, the window size W is set to $2E$, and in the second case, it is set to $4|y_s|$. The sweep counter is set to 1 and the sweep direction is set to north. The robot continues east from where it arrives. (See Figure 4.2(b).)

Rule 3: Otherwise, do the following. Without loss of generality, we assume that the current sweep direction is south. If $E = \min(E, N, S)$, the robot goes through the obstacle to point $(x+w, y)$, and then travels south until it arrives at

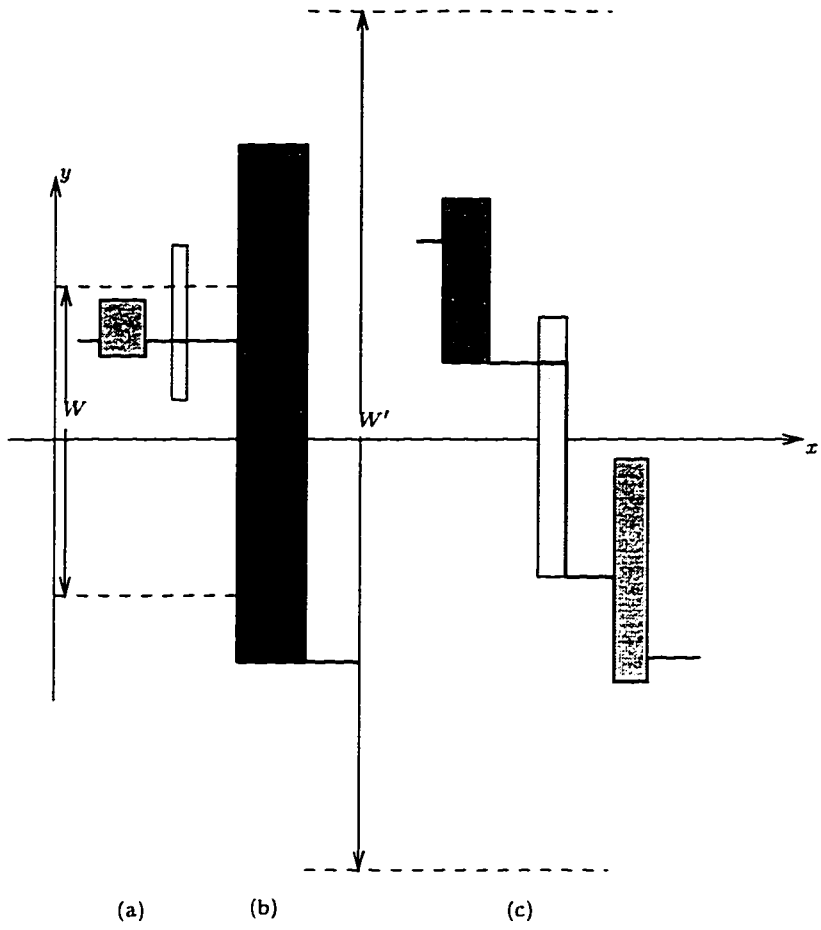

Figure 4.2: The three rules of the generalized sweeping strategy.

$(x+w, \max(y_s, -W/2))$. Otherwise, it goes along the boundary of the obstacle, through the nearest corner, until it arrives at $(x+w, \max(y_s, -W/2))$. In both cases, if the robot arrives at $(x+w, -W/2)$, the sweep counter is incremented by 1 and the sweep direction is flipped to north. The robot continues east from where it arrives. (See Figure 4.2(c).)

4.1.2 Analysis

Lemma 4.1.1 *The total effort spent by the robot is $O(W_f\sqrt{n})$, where W_f is the final window size.*

Proof: We divide the effort into three components: (1) the effort of traveling along the horizontal segments in the free space, (2) the effort of traveling using Rule 1, and (3) the effort of traveling using the other rules, and bound each part separately. (1) Since the robot never backtracks horizontally, the effort of traveling along the horizontal segments in the free space is bounded by $O(n)$. (2) Let $\tau_f = W_f/\sqrt{n}$ be the largest threshold. Since the width of each obstacle is at least 1, the effort of traveling using Rule 1 is bounded by $O(n\tau_f) = O(W_f\sqrt{n})$. (3) To prove the lemma, it suffices to bound this component by $O(W_f\sqrt{n})$.

Fix a window size W, in one sweep, the effort of traveling using Rule 2 or 3 is bounded by $O(W)$. This is obviously true if the robot uses Rule 2. Otherwise, assume that the sweep direction is south. We can further decompose the effort into two parts: the forward-going effort – the effort of making progress toward south, and the backtracking effort, which includes the effort of going through obstacles and the effort of going around a north corner and arriving at a point which has the same y-coordinate as the point where the robot hits the obstacle. The total forward-going effort in a sweep is $O(W)$. If we can show that the backtracking effort can be charged to the forward-going effort by a constant factor, then the total effort spent in a sweep is bounded by $O(W)$. We will show for the case when the robot goes through an obstacle using Rule 3. The other cases can be dealt with in a similar manner. If $y_s > -W/2$, the robot travels south to point $(x + w, y_s)$ after it goes through the obstacle to point

$(x+w, y)$. Thus after if spends an effort of E, it makes progress toward south by an amount of S. Since $E < S$ (that's why the robot chooses to go through the obstacle), we can charge E to S. If $y_s < -W/2$, the robot travels south to point $(x+w, -W/2)$ after it goes through the obstacle to point $(x+w, y)$. In this case, E may be larger than the effort of progressing toward south. But notice that $E < N < W$ (otherwise the robot has to use Rule 2), and this case happens only once in a sweep. Thus the total effort of going through obstacles in a sweep is bounded by $O(W)$.

Since the robot does at most $\sqrt{n}$ sweeps for each window size W, the total effort spent for a fixed window size is $O(W\sqrt{n})$. The window size is at least doubled each time it is changed, thus the total effort for traveling using Rule 2 and Rule 3 is also $O(W_f\sqrt{n})$. This completes the proof of the lemma. □

Lemma 4.1.2 *The minimum effort required to travel from s to t is $\Omega(W_f)$, where W_f is the final window size.*

Proof: Since the Euclidean distance from s to t is n, the minimum effort is obviously $\Omega(n)$. If $W_f \leq n$, then we are done. Otherwise, we prove that the minimum effort is $\Omega(W_f)$.

Suppose that the robot has completed $\sqrt{n}$ sweeps for some window sizes, and that W is the largest such window size. We claim that the minimum effort is $\Omega(W)$. The minimum-effort path may either be going a vertical distance of $\Omega(W)$ to go around all the obstacles, or be cutting through each sweep. The minimum effort of cutting through one sweep is $\Omega(\tau)$, where $\tau = W/\sqrt{n}$. To see this, let's consider the obstacles met by the robot during one sweep. Without

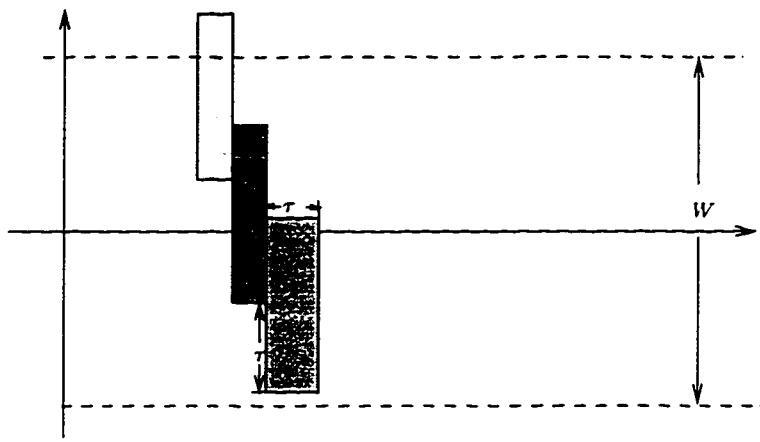

Figure 4.3: To cut through one sweep, the minimum effort is $\Omega(\tau)$.

loss of generality, we can assume that the sweep direction is south. If we omit all the obstacles to which the robot applied Rule 1, and omit the horizontal free space between two successive obstacles met by the robot, then we obtain a "barrier" similar to the one shown in Figure 4.3. Initially, from the north window side, to overcome the first obstacle in the sweep takes an effort of at least τ. Then from the south-east corner of an obstacle, to overcome the next obstacle takes an effort of at least τ. Otherwise, the robot should have applied Rule 1 to the obstacle. This barrier stretches at least τ over both window sides. To cut through this sweep, the minimum-effort path needs to overcome one of these obstacles. However, to overcome any one of them takes at least τ effort. Thus the effort of cutting through all the $\sqrt{n}$ sweeps is $\Omega(\sqrt{n}\tau) = \Omega(W)$.

If $W_f \leq 2W$ then we are done. Otherwise (i.e. $W_f > 2W$ or there is no window for which the robot has completed $\sqrt{n}$ sweeps), then it must be the case that W_f is determined by Rule 2, in which case the minimum effort is clearly $\Omega(W_f)$. □

Combining Lemma 4.1.1 and Lemma 4.1.2, we obtain

Theorem 4.1.3 *The competitive ratio of the generalized sweeping strategy is bounded by* $O(\sqrt{n})$.

4.1.3 The lower bound

Lemma 4.1.4 *Any on-line algorithm for the wall problem with penetrable obstacles has a competitive ratio of* $\Omega(\sqrt{n})$.

Proof: We prove the lemma by an "adversary" argument, similar to the one given in [81]. The adversary builds a scene from a set of identical obstacles. Each obstacle has a height of n, a width of 1, and a density of n.

Once the robot overcomes an obstacle, the adversary always puts another obstacle right in front of the robot, such that the robot is in the middle of the obstacle's height. Thus no matter which way the robot chooses to overcome an obstacle, the effort is $\Omega(n)$. Since there are n obstacles, the total effort spent by the robot to get to the goal is $\Omega(n^2)$.

On the other hand, by the pigeon hole theorem, either all the obstacles lie within the strip bounded by $y = \pm n^{3/2}$, or there is a line $y = k$, where $-n^{3/2} \leq k \leq n^{3/2}$, such that the line passes at most $n^{1/2}$ obstacles. In the first case, the robot can go a vertical distance of $n^{3/2}$ to go around all the obstacles. In the second case, the robot follow the line $y = k$, encountering $\sqrt{n}$ obstacles and spending a total effort of $O(n^{3/2})$ to overcome them. Either way, the effort spent by the adversary is at most $O(n^{3/2})$.

Thus the competitive ratio is lower bounded by $O(\sqrt{n})$. □

4.2 The Recursive Wall Problem

4.2.1 The model

We extend the model of the wall problem with penetrable obstacles to include penetrable obstacles which contain obstacles of higher densities within.

The *depth* of an obstacle is defined recursively as follows. An obstacle which is placed immediately inside the free space has a depth of 0. An obstacle which is placed immediately inside an obstacle of depth i has a depth of $i + 1$. The *level of recursion* of a scene is R if the depth of the deepest obstacle is R. If an obstacle has a width of w and a density of d, then the *expanded Euclidean distance* of this obstacle is dw.

We have made the assumption that the width of each obstacle is at least 1 in the Wall Problem with Penetrable Obstacles. Here we make a similar assumption as follows. If an obstacle O_1 with a width of w_1 and a density of d_1 is placed within another obstacle of width w_2 and density d_2, then $w_1 d_2$ is at least 1.

In the following, we let n denote the Euclidean distance from the source point to the wall. And let n_i denote the upper bound on the expanded Euclidean distances of the obstacles of depth i.

The following section gives a lower bound on the Recursive Wall Problem when $R = 1$. The result can be generalized to higher recursive level. The difficulty in giving an upper bound is described in Section 4.3.

4.2.2 The lower bound

Lemma 4.2.1 *When $R = 1$, the lower bound of the competitive ratio is*

$$\begin{cases} {n_0}^{1/2}{n_0}^{1/4} & \text{if } n_0 \leq n^2, \\ {n_0}^{1/2} & \text{otherwise.} \end{cases}$$

Proof: The adversary can build a scene using a set of identical 0-level obstacles. Each obstacle has a height of n_0^2, a width of 1, but a density of n_0 (thus the expanded Euclidean distance of each 0-level obstacle is n_0). Using a construction similar to the proof of the lower bound in Lemma 4.1.4, the adversary can construct 1-level obstacles within a 0-level obstacle in such a way that the minimum effort of going through a 0-level obstacle is $O({n_0}^{3/2})$, while any on-line algorithm has to spend an effort of $\Omega({n_0}^2)$ (notice that this is possible even though the height of the 0-level obstacle is limited - a point different from the situation in Lemma 4.1.4).

No matter which way the robot chooses to overcome a 0-level obstacle, the adversary puts another 0-level obstacle immediately in front of it. Thus the total effort the robot has to spent is $\Omega(n{n_0}^2)$.

By the pigeon hole theorem, the optimal path may be either going a vertical distance of $n{n_0}^2/x$ to go around all the obstacles, spending an effort of $O(n{n_0}^2/x)$, or following a path that hits at most x obstacles, spending an effort of $O({n_0}^{3/2})$ on each and $O(x{n_0}^{3/2})$ in total. The maximum of $\min(x{n_0}^{3/2}, n{n_0}^2/x)$ is achieved when $x = n^{1/2}{n_0}^{1/4}$. Since $1 \leq x \leq n$, when $n_0 \leq n^2$, $x = n^{1/2}n_0^{1/4}$ and when $n_0 > n^2$, $x = n$. The effort spent by the adversary is $O(n^{1/2}{n_0}^{7/4})$, when $n_0 \leq n^2$, and $O(n{n_0}^{3/2})$ when $n_0 > n^2$. Thus the competitive ratio is lower bounded by $\Omega(n^{1/2}{n_0}^{1/4})$ when $n_0 \leq n^2$, and $\Omega({n_0}^{1/2})$ otherwise. □

4.3 Discussions

If first seems that we can give an upper bound for the recursive wall problem, by applying the generalized sweeping strategy recursively to overcome each 0-level obstacles. We need to guarantee that for each 0-level obstacle, the robot spends an effort no more than $O(\sqrt{n_0})$ times the optimal effort. However, the upper bound of $O(\sqrt{n_0})$ applies only when the robot and the adversary enter the obstacle at the same point. If they enter an obstacle at different points, it is obvious that the adversary can arrange the 1-level obstacles within this obstacle in such a way that it takes the robot much more effort to go through than it does the adversary. Because of this difficulty, we can not obtain an upper bound by applying the generalized sweeping strategy.

We made a strong assumption at the very beginning that the robot knows the shape, the size, the position and the density of an obstacle the first time it encounters it. We can essentially achieve the same upper bound for the wall problem with penetrable obstacles (up to a constant factor) under a much weaker assumption. We can assume that the robot uses only *tactile* sensors, i.e., the robot learns about the obstacles only by touching them and moving along them. The generalized sweeping strategy essentially requires that upon encountering an obstacle, the robot can figure out the smallest of the three quantities E, N and S by spending an effort within a constant factor of $\min(E, N, S)$. We will show that a tactile robot can achieve this by using the *doubling strategy* of Baeza-Yates *et al.* [5].

The above problem can be generalized to the following *cow-path* problem.

Consider a cow standing at a crossroad with d paths leading off into an unknown territory. On one of the paths there is a grazing field (the target t) at distance n from the intersection, and all the paths go on forever. The cow won't know that it has found the field until it touches it. The problem is to reach the target while traveling the least distance possible. The doubling strategy works as follows: Try the d paths in turns, and each time, double the length of trial. So the cow follows path 1 for a unit length. If the field is not reached, it backtracks along the path to the intersection and then follow path 2 for 2-unit length. Then it follows path 3 for 4-unit length, and so on. If the field is not reached after following 2^{d-1}-unit length along path d, it tries path 1 for 2^d-unit length and so on. The competitive ratio of this strategy is $\frac{2d^d}{(d-1)^{d-1}} + 1$, and is optimal for deterministic algorithms. When d is fixed, the ratio is a constant. Kao *et al.* [52] developed a randomized algorithm for this problem. Their expected performance was shown to be twice as good as the deterministic one and is optimal for $d = 2$. They conjecture that the algorithm is also optimal for $d = 3$.

We can also extend the wall problem with penetrable obstacles and the recursive wall problem to 3 dimensions, where the target t is an infinite plane perpendicular to the x-axis, and the obstacles are oriented cuboids. Using similar arguments, we can show that the lower bound on the competitive ratio is $\Omega(n^{2/3})$ for WPPO, and $\Omega(n^{2/3}{n_0}^{1/3})$ for the recursive wall problem with recursion level of 1, where n is the Euclidean distance between the start point and the target plane, and n_0 is the upper bound on the expanded Euclidean distance. We can generalize the *spiral strategy* of [11] to obtain an optimal

upper bound of $\Theta(n^{2/3})$ for WPPO. Briefly, the spiral strategy is similar to the sweeping strategy, but instead of sweeping upwards and downwards, the robot spirals outwards and inwards. The robot still keeps four variables: the window size W, the threshold τ, the spiral direction, and the counter. W is initialized to be n, while τ is always set to be $W/n^{1/3}$. The spiral has its center on x axis and its orbits lie in a plane perpendicular to the x-axis. The spacing between adjacent orbits of the spiral is always $\leq \tau/3$. The yz coordinates of the robot will always lie on a spiral. The spiral direction is either outwards or inwards. If the robot completes $n^{1/3}$ spirals for a fixed window size, the window size is doubled. The robot applies rules similar to the rules for the 2-dimensional case.

There are some other variations of the wall problem with penetrable obstacles. One is to allow obstacles with densities smaller than 1. Such obstacles model rivers where the robot can take advantage of the flow. Another variation is to allow flows within obstacles. The directions of the flows are along y-axis and the robot can be carried away by the flow while it is exploring the width of the obstacle. Assuming that the flow has a constant velocity which can be learned by the robot as soon as it touches the obstacle, we don't know whether the sweeping strategy can be applied.

Chapter 5

Conclusion

In this dissertation, we have presented approximate and adaptive algorithms for some optimal motion-planning problems. We conclude this dissertation by suggesting some open problems for future research.

In the kinodynamic motion-planning problem, we only considered *Cartesian robots*, i.e., robots whose inertia tensor is constant (see [34] for a more precise definition). We believe that our approach can also be applied to more general robots with *open kinematic chains*. In order to do this, we need to generalize our Correcting Lemma to open chains. A key further problem is to determine cases of kinodynamic motion-planning problems for which we can solve in closed forms, or give sufficient characterizations for developing faster algorithms. An example along this line is the case of planning for a point robot moving on a plane amid polygonal obstacles and with decoupled kinodynamic constraints. It is characterized in [18] that the minimum-time trajectory is a sequence of segments, where each segment is a "bang-bang" control between two obstacle boundary points. It is interesting to investigate whether such charac-

terizations extend to coupled case and higher dimensions. The complexity of the kinodynamic motion-planning problem in 2D remains open.

Our algorithm for the 2D curvature-constrained shortest-path problem (see Section 3.3) may be further improved, in two ways. First, the running time may be improved from $O(n^2)$ to subquadratic. In our algorithm, we consider all the edges of the graph G. Since the number of edges is $\Omega(n^2)$ in the worst case, the running time is $\Omega(n^2)$. But since we are only interested in computing approximate shortest paths, it may be sufficient to construct a small subset of the edges of G, whose size is linear. Such techniques have been used to compute approximate unconstrained shortest paths amid obstacles in [24]. Second, the running time may be improved from $O((1/\varepsilon)^4)$ to $O((1/\varepsilon)^3)$ or even better. This improvement may result from a non-uniform discretization, similar to the one used in our approximation algorithm for the kinodynamic motion-planning problem. Roughly speaking, the fineness of discretization need not be constant, but should be proportional to the lengths of the Dubins paths.

In the curvature-constrained shortest-path problem, the robot is restricted to be a point. An interesting question is whether our characterizations and algorithms can be generalized to disc or polygon robots. The complexity of the curvature-constrained shortest-path problem in 2D is an intriguing open question. Our approximation algorithm for this problem in 3D does not use Sussman's characterization [99] of such paths in 3D. We expect that faster approximation algorithms can be developed by utilizing this characterization.

We have learned two majors lessons from our study of constrained motion-planning problems, which can be our guidelines in future research. One is

that in developing efficient algorithms for computing constrained paths, it is important to exploit the geometry of such paths. The second lesson is to exploit the power of non-uniform discretization in developing fast approximation algorithms.

As to the on-line navigation problems, current work is restricted to obstacles with very simple geometries. It is not know whether strategies like the sweeping strategy can be generalized to deal with general polygons. It is also an interesting problem to consider the room problem with penetrable obstacles. If we can solve this problem efficiently, then we can solve point-to-point navigation with penetrable obstacles, by combining the strategies for the wall problem and the room problem. An interesting variation of the penetrable obstacles is to allow the densities to be less than 1 (but larger than 0). This models the situation where the robot can take advantage of a river flow.

There is very little known about on-line motion planning with dynamic constraints.

Appendix A

The TC-Graph Method

For the sake of completeness, we describe the gists of the Tracking Lemma and the TC-graph method in this section.

A trajectory Γ' is said to *track* another trajectory Γ to a tolerance (η_x, η_v), if for all $t \in [0, T]$,

$$\|p_\Gamma(t) - p_{\Gamma'}(t)\|_\infty \leq \eta_x, \text{ and} \tag{A.1}$$

$$\|v_\Gamma(t) - v_{\Gamma'}(t)\|_\infty \leq \eta_v. \tag{A.2}$$

Notice that Equation A.1 implies that $d(\Gamma', \Gamma) \leq \eta_x$.

A τ-*bang* is a trajectory segment of time duration τ, during which a *constant* acceleration is applied[1]. A τ-*bang trajectory* is a trajectory consisting of a sequence of τ-bangs. The set of bang trajectories is a restricted set of trajectories. But they suffice to track any non-restricted trajectories, if the acceleration set used by the bang trajectories has some *advantage* over the acceleration set

[1]Here we slightly abuse the notion of a *bang*. A bang usually means that its acceleration should be *extremal* in some sense.

used by the non-restricted trajectories. Next we define the notion of advantage more formally.

Let $\mathcal{P}$ and $\mathcal{Q}$ be two sets of accelerations. For any acceleration $a \in \mathcal{P}$, and any direction given by a d-length vector σ of 1's and -1's, if there exists an acceleration $b \in \mathcal{Q}$, such that

$$\sigma^j(b^j - a^j) \geq \kappa_l, \tag{A.3}$$

for $1 \leq j \leq d$, then $\mathcal{Q}$ is said to have a *uniform κ_l advantage* over $\mathcal{P}$ (α^j means the jth element of a vector α). The Tracking Lemma relates the parameter τ to the uniform advantage κ_l and the tracking tolerance (η_x, η_v).

Lemma A.0.1 (Tracking Lemma [34]) [2] *Let $\mathcal{P}$ and $\mathcal{Q}$ be two sets of accelerations such that, for each $a \in \mathcal{Q}$, $\|a\| \leq 1$, and that $\mathcal{Q}$ has a uniform κ_l advantage over $\mathcal{P}$.*

Let Γ be a trajectory that uses $\mathcal{P}$. Let (η_x, η_v) be a tracking tolerance. There exists a time step τ, a τ-bang trajectory Γ' that uses $\mathcal{Q}$, such that Γ' tracks Γ to tolerance (η_x, η_v).

Moreover, it is sufficient that

$$\tau = O(\min(\eta_v, \sqrt{\eta_x \kappa_l})). \tag{A.4}$$

Let $\mathcal{A}$ (resp. $\mathcal{A}_\varepsilon$) be the set of accelerations whose L_2 norms are bounded by 1 (resp. $1/(1+\varepsilon)^2$). Next we show how we can choose a finite set of

[2] This is a simplified version of the Tracking Lemma presented in [34]. In [34], $\mathcal{Q}$ is a set of *instantaneous accelerations* that depends on the current state, and may be a subset of the set of allowed accelerations. This is because applying certain accelerations may violate the velocity constraint. In the context where we apply this Tracking Lemma, we do not explicitly bound the velocity of the tracking trajectory. This means that the set of instantaneous accelerations equals the set of all allowed accelerations.

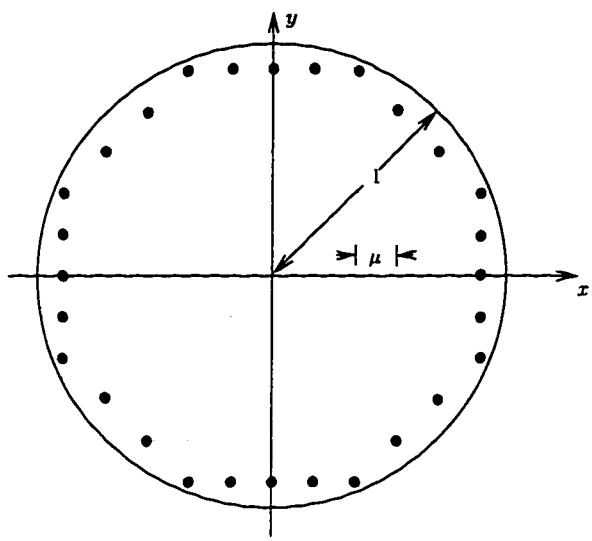

Figure A.1: The acceleration set $\mathcal{A}_\mu$ in 2D.

accelerations such that this set has a uniform advantage over $\mathcal{A}_\varepsilon$. Given a parameter μ, a set of gridpoints of $\mathcal{A}$ with spacing μ is defined to be

$$\{a = (i_1\mu, \ldots, i_d\mu) \mid i_1, \ldots, i_d = 0, \pm 1, \pm 2, \ldots, a \in \mathcal{A}\}. \tag{A.5}$$

Let $\mathcal{A}_\mu$ be the set of *boundary* gridpoints[3] (see Figure A.1; the dark dots represent the boundary gridpoints in 2D). It is shown in [34], that if μ is chosen such that

$$\mu \leq \kappa_l \leq \frac{\varepsilon}{4\sqrt{d}}, \tag{A.6}$$

then $\mathcal{A}_\mu$ has a uniform advantage of κ_l over $\mathcal{A}_\varepsilon$. Notice that $|\mathcal{A}_\mu| = O(d(1/\mu)^{d-1})$.

A τ-bang (resp. τ-bang trajectory) is a (μ, τ)-*bang* (resp. (μ, τ)-*bang trajectory*) if the acceleration set used is $\mathcal{A}_\mu$. Given a tracking tolerance (η_x, η_v),

[3] In [34], two approximation algorithms, the True-Extremal algorithm and the Near-Extremal algorithm are described. They differ in the way of choosing a discretized acceleration set, and in the way of defining bangs. The description here follows from the Near-Extremal algorithm.

and any $\varepsilon > 0$, there exist μ (satisfying Equation A.6) and τ (satisfying Equation A.4), such that for any $(1/(1+\varepsilon)^2, 1/(1+\varepsilon))$-trajectory Γ, there exists a (μ, τ)-bang trajectory Γ' (also a $(1,1)$-trajectory) that tracks Γ to a tolerance of (η_x, η_v). To approximate any given $(1,1)$-trajectory Γ to within a factor of $(1+\varepsilon)$ in time length, we first time-rescale Γ to a $(1/(1+\varepsilon)^2, 1/(1+\varepsilon))$-trajectory Γ', with time length extended by a factor of $(1+\varepsilon)$. Since there exists a (μ, τ)-bang trajectory Γ'' that tracks Γ' to a tolerance of (η_x, η_v), Γ'' approximates Γ in that $T(\Gamma'') \leq (1+\varepsilon)\, T(\Gamma)$. But Γ'' does not track Γ to the tolerance of (η_x, η_v). However, $d(\Gamma'', \Gamma) \leq \eta_x$. This is because Γ' and Γ trace the same path and $d(\Gamma'', \Gamma') \leq \eta_x$. Also the two end states of Γ'' are close to the two end states of Γ respectively.

Having known the existence of a tracking bang trajectory, the next question is how to one. What makes it more difficult is that most of the time, the original $(1,1)$-trajectory Γ is not given except its initial and final states. The TC-graph method transforms this problem to that of finding a shortest path in a directed graph. (In the following, we only describe the TC-graph method in absence of obstacles, where we do not have to do collision-free checking.) The TC-graph G is roughly as follows: (a) The *root node* of G approximates the initial state of Γ; (b) A directed edge corresponds to a (μ, τ)-bang whose velocities are also bounded by 1 in L_2 norm. The weight of the edge is always τ; (c) The graph is generated and explored from the root node in a breadth-first manner, and the search terminates when either a node approximating the final state is found, or when no new nodes are generated. Notice that

- The size of G is finite (see the following analysis).

- The breadth-first search produces a trajectory whose time length is no more than $(1+\varepsilon)\,T(\Gamma)$. The breadth-first search suffices for finding a shortest path since each edge has a uniform weight of τ.

- Suppose that Γ lies within a subset W of the configuration space. Since there exists a bang trajectory Γ'' satisfying $d(\Gamma'', \Gamma') \leq \eta_x$, then $d(\Gamma'', W) \leq \eta_x$. We can find such a trajectory by considering only those edges whose corresponding bangs do not diverge from W by more than η_x. Notice that it still holds that $T(\Gamma'') \leq (1+\varepsilon)\,T(\Gamma)$.

Each node of G corresponds to a state. By carefully choosing the initial velocity, we can bound the number of possible velocities by $O((1/(\mu\tau))^d)$, and the number of possible locations by $O((L/(\mu\tau^2))^d)$, where L is the size of W. Thus the total number of nodes is bounded by

$$O\left(\left(\frac{L}{\mu^2\tau^3}\right)^d\right). \tag{A.7}$$

Each node can have at most $O(d(1/\mu)^{d-1})$ out-going edges (because it can have at most this many choices of accelerations), thus the total number of graph edges is

$$O\left(\frac{dL^d}{\mu^{3d-1}\tau^{3d}}\right). \tag{A.8}$$

This also bounds the running time.

If we set

$$\eta_x = \varepsilon\rho/2, \text{ and } \eta_v = \varepsilon/2, \tag{A.9}$$

where $\rho = O(1)$, we obtain Corollary 2.2.2. Notice that since $\tau = O(\varepsilon)$ and $\mu = O(\varepsilon)$, the running time is $O(L^d(1/\varepsilon)^{6d-1})$.

Bibliography

[1] P.K. Agarwal and J. Matousek. On range searching with semialgebraic sets. In *Discrete Comput. Geom.*, 11:393–418, 1994.

[2] P.K. Agarwal, P. Raghavan, and H. Tamaki. Motion planning for a steering-constrained robot through moderate obstacles. In *Proc. 27th Symp. on Theory of Computing*, pages 343–352, 1995.

[3] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[4] T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility polygon search and Euclidean shortest paths. In *Proc. 26th Symp. on Foundations of Computer Science*, pages 155–164, 1985.

[5] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106:234–252, 1993.

[6] E. Bar-Eli, P. Berman, A. Fiat, and P. Yan. On-line navigation in a room. In *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms*, pages 237–249, 1992.

[7] J. Barraquand and J.C. Latombe. A Monte-Carlo algorithm for path planning with many degrees of freedom. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1712–1717, 1990.

[8] J. Barraquand and J.C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[9] J. Bentley and T. Ottmann. Algorithms for reporting and counting intersections. *IEEE Trans. Comput.*, 28:643–647, 1979.

[10] P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosen, and M. Saks. Randomized robot navigation algorithms. In *Proc. 7th ACM-SIAM Symp. on Discrete Algorithms*, pages 75–84, 1996.

[11] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. In *Proc. 23rd Symp. on Theory of Computing*, pages 494–504, 1991.

[12] J.-D. Boissonnat and S. Lazard. A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles. In *Proc. 12th Symp. on Computational Geometry*, pages 242–251, 1996.

[13] J.D. Boissonnat, A. Cerezo, and J. Leblond. Shortest paths of bounded curvature in the plane. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2315–2320, 1992.

[14] J. Canny. Private communication.

[15] J. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[16] J. Canny. Some algebraic and geometric configurations in PSPACE. In *Proc. 20th Symp. on Theory of Computing*, pages 460–467, 1988.

[17] J. Canny, B. Donald, J. Reif, and P. Xavier. On the complexity of kinodynamic planning. In *Proc. 29th Symp. on Foundations of Computer Science*, pages 306–316, 1988.

[18] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynomic planning in the plane. *Discrete Comput. Geom.*, 6:461–484, 1991.

[19] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th Symp. on Foundations of Computer Science*, pages 49–60, 1987.

[20] K. Chan and T. Lam. An on-line algorithm for navigating in unknown terrain. ?

[21] L.P. Chew. Planning the shortest path for a disc in $O(n^2 \log n)$ time. In *Proc. 1st ACM Symp. on Computational Geometry*, pages 214–233, 1985.

[22] J. Choi, J. Sellen, and C.K. Yap. Approximate Euclidean shortest path in 3-space. In *Proc. 10th Ann. Symp. on Computational Geometry*, pages 41–48, 1994.

[23] J. Choi, J. Sellen, and C.K. Yap. Precision-sensitive Euclidean shortest path in 3-space. In *Proc. 11th Symp. on Computational Geometry*, pages 350–359, 1995.

[24] K.L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 9th Symp. on Theory of Computing*, pages 56–65, 1987.

[25] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *J. Symbolic Comput.*, 7:11–30, 1989.

[26] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* The MIT Press, 1991.

[27] A. Datta and C. Icking. Competitive searching in a generalized street. In *Proc. 10th Symp. on Computational Geometry*, pages 175–182, 1994.

[28] P.J. de Rezende, D.T. Lee, and Y.F. Wu. Rectilinear shortest paths in the presence of rectangular barriers. *Discrete Comput. Geom.*, 4:41–53, 1989.

[29] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 298–303, 1991.

[30] B. Donald and P. Xavier. Near-optimal kinodynamic planning for robots with coupled dynamics bounds. In *IEEE Int. Symp. on Intelligent Controls*, pages 354–359, 1989.

[31] B. Donald and P. Xavier. A provable good approximation algorithm for optimal-time trajectory planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 958–963, 1989.

[32] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators. In *Proc. 6th Symp. on Computational Geometry*, pages 290–300, 1990.

[33] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14:443–479, 1995.

[34] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators. *Algorithmica*, 14:480–530, 1995.

[35] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *J. of the ACM*, 40:1048–1066, 1993.

[36] L.E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *Amer. J. Math.*, 79:497–516, 1957.

[37] B. Faverjon and P. Tournassound. A local approach for path planning of manipulators with a high number of degrees of freedom. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1152–1159, 1987.

[38] B. Faverjon and P. Tournassound. A practical approach to motion planning for manipulators with many degrees of freedom. In *Proc. 5th Int. Symp. on Robotics Research*, pages 65–73, 1989.

[39] S. Fortune and G. Wilfong. Planning constrained motion. *Annals of Mathematics and Artificial Intelligence*, 3:21–82, 1991.

[40] S.K. Ghosh and D.M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM J. Comput.*, 20(5):888–910, 1991.

[41] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R.E. Tarjan. Linear time algorithms for shortest path and visibility problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[42] L. Guibas, M. Overmars, and M. Sharir. Ray shooting, implicit point location, and related queries in arrangements of segments. Technical report, Dept. Computer Science, New York University, 1989.

[43] G. Heinzinger, P. Jacobs, J. Canny, and B. Paden. Time-optimal trajectories for a robot manipulator: A provably good approximation algorithm. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 150–155, 1990.

[44] J. Hershberger and L. Guibas. An $O(n^2)$ shortest path algorithm for a non-rotating convex body. *J. Algorithms*, 9:18–46, 1988.

[45] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. In *Lecture Notes in Computer Science*, volume 519, pages 331–342. 1991.

[46] J. Hershberger and S. Suri. Efficient computation of Euclidean shortest paths in the plane. In *Proc. 34th Symp. on Foundations of Computer Science*, pages 508–517, 1993.

[47] J.M. Hollerbach. Dynamic scaling of manipulator trajectories. In *Proc. of the Automatic Control Council*, pages 752–756, 1983.

[48] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In Z. Li and J. Canny, editors, *Nonholonomic Motion Planning*, pages 271–342. Kluwer Academic Publishers, 1992.

[49] B. Kalyanasundaram and K. Pruhs. Visual searching and mapping. In *DIMACS Workshop on Online Algorithms*, 1991.

[50] B. Kalyanasundaram and K. Pruhs. A competitive analysis of algorithms for searching unknown scenes. *Comput. Geom. Theory Apply.*, 3(3):139–155, 1993.

[51] B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. In *Proc. 20th Int. Colloquium on Automata, Languages and Programming*, pages 102–113, 1993.

[52] M. Kao, J. Reif, and S. Tate. Searching in an unknown environment: an optimal randomized algorithm for the cow-path problem. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, 1993.

[53] S. Kapoor and S.N. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proc. 4th Symp. on Computational Geometry*, pages 172–182, 1988.

[54] H. Karloff, Y. Rabani, and Y. Ravid. Lower bounds for randomized k-server and motion-planning algorithms. *SIAM J. on Computing*, 23:293–312, 1994.

[55] L. Kavraki and J.C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2138–2145, 1994.

[56] L. Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. In *Proc. 27th Symp. on Theory of Computing*, 1995.

[57] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[58] R. Klein. Walking in an unknown street with bounded detour. *Comput. Geom. Theory and Apply*, 1:325–351, 1992.

[59] J. Kleinberg. On-line search in a simple polygon. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 8–15, 1994.

[60] V.P. Kostov and E.V. Degtiariova-Kostova. Suboptimal paths in the problem of a planar motion with bounded derivative of the curvature. Technical report, INRIA, Cedex (France), 1993.

[61] V.P. Kostov and E.V. Degtiariova-Kostova. The planar motion with bounded derivative of the curvature and its suboptimal paths. Technical report, INRIA, Cedex (France), 1994.

[62] J.C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, MA, 1991.

[63] J.P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *Intelligent Autonomous Systems.* North Holland, 1987.

[64] J.P. Laumond, P.E. Jacobs, M. Taix, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. on Robotics and Automation*, 10(5):577–593, 1994.

[65] J.P. Laumond and T. Simeon. Motion planning for a two degrees of freedom mobile robot with towing. Technical report, LAAS/CNRS, LAAS, Toulouse, France, 1989.

[66] D.T. Lee and F.P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

[67] Z. Li and J.F. Canny. Robot motion planning with non-holonomic constraints. Technical report, Electronics Research Lab, Univ. of California, Berkeley, 1989.

[68] Z. Li, J.F. Canny, and S.S. Sastry. On motion planning for dextrous manipulation. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1990.

[69] G.L. Miller, D. Talmor, S. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: generation, formulation, and

partition. In *Proc. 27th Symp. on Theory of Computing*, pages 683–692, 1995.

[70] G.L. Miller, S. Teng, W. Thurston, and S.A. Vavasis. Automatic mesh partitioning. In A. George *et al.*, editor, *Graphs Theory and Sparse Matrix Computation*, pages 57–84. Springer-Verlag, 1993.

[71] B. Mirtich and J. Canny. Using sekletons for nonholonomic path planning among obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Autonomation*, pages 2533–2540, 1992.

[72] J.S.B. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3:83–106, 1991.

[73] J.S.B. Mitchell. Shortest paths among obstacles in the plane. In *Proc. 9th Symp. on Computational Geometry*, 1993.

[74] J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitrious. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, 1987.

[75] J.S.B. Mitchell, D.M. Mount, and S. Suri. Query-sensitive ray shooting. In *Proc. 10th Symp. on Computational Geometry*, pages 359–368, 1994.

[76] J.S.B. Mitchell and C. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. Technical report, Cornell Univ., 1985.

[77] D. Moore. The cost of balancing generalized quad-tree. Manuscript, 1995.

[78] C. Ó'Dúnlaing. Motion planning with inertial constraints. *Algorithmica*, 2(4):431–475, 1987.

[79] M.H. Overmars and P. Svestka. A probabilistic learning approach to motion planning. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, pages 19–37, 1994.

[80] M.H. Overmars and E. Welzl. New methods for computing visibility graph. In *Proc. 4th Symp. on Computational Geometry*, pages 164–171, 1988.

[81] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Proc. 16th Int. Colloquium on Automata, Languages and Programming*, pages 610–620, July 1989.

[82] C.H. Papadimitriou. An algorithm for shortest path motion in three dimensions. *Information Processing Letters*, 20:259–263, 1985.

[83] F. Preparata. A note on locating a set of points in a planar subdivision. *SIAM J. Comput.*, 8:542–545, 1979.

[84] J.A. Reeds and L.A. Shepp. Optimal paths for a car that goes both forfards and backwards. *Pacific J. of Mathematics*, 145(2):367–393, 1990.

[85] J. Reif. Complexity of the generalized movers problem. In J. Hopcorft, J. Schwartz, and M. Sharir, editors, *Planning, Geometry and Complexity*

of Robot Motion, pages 267–281. Ablex Pub. Corp., Norwood, NJ, 1987.

[86] J. Reif and J.A. Storer. Shortest paths in Euclidean space with polyhedral obstacles. Technical report, Computer Science Dept., Brandeis Univ., 1985.

[87] J. Reif and S. Tate. Continuous alternation: the complexity of pursuit in continuous domains. *Algorithmica*, 10:156–181, 1993.

[88] J. Reif and S.R. Tate. Approximate kinodynamic planning using L_2-norm dynamic bounds. *Computers Math. Applic.*, 27(5):29–44, 1994.

[89] J. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. In *Proc. Workshop on the Algorithmic Foundations of Robotics*, pages 331–345, 1994.

[90] G. Sahar and J.M. Hollerbach. Planning of minimum-time trajectories for robot arms. Technical report, MIT, 1984.

[91] J.T. Schwartz and M. Sharir. On the piano movers' problem I: The special case of two-dimensional rigid polygonal body moving amidst polygonal barriers. *Comm. Pure and Appl. Math.*, 36:345–398, 1983.

[92] J.T. Schwartz and M. Sharir. On the piano movers' problem II: General techniques for computing topological properties of real algebraic manifolds. *Adv. in Appl. Math*, 4:298–351, 1983.

[93] J.T. Schwartz and M. Sharir. On the piano movers' problem: III. coordinating the motion of several independent bodies: The special case

of circular bodies moving amidst polygonal barriers. *Int. J. of Robotics Research*, 2(3):46–75, 1983.

[94] J.T. Schwartz and M. Sharir. Motion planning and related geometric algorithms in robotics. In *Proc. of the Int. Congress of Mathematicians*, pages 1594–1611, 1986.

[95] J.T. Schwartz and M. Sharir. Algorithmic motion planning in robotics. In J.V. Leeuwen, editor, *Algorithms and Complexity, Volume A of Handbook of Theoretical Computer Science*, chapter 8, pages 391–430. Elsevier, Amsterdam, 1990.

[96] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15:193–215, 1986.

[97] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28:202–208, 1985.

[98] M. Spivak. *A Comprehensive Introduction to Differential Geometry.* Publish or Perish, 1979.

[99] H.J. Sussmann. Shortest 3-dimensional paths with a prescribed curvature bound. In *Proc. 34th Conf. on Decision and Control*, pages 3306–3312, 1995.

[100] Z. Vafa and S. Dubowsky. On the dynamics of manipulators in space using the virtual manipulator approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1987.

[101] E. Welzl. Constructing the visibility graph for n line segments in $O(n^2)$ time. *Inform. Process. Lett.*, 20:167–172, 1985.

[102] G. Wilfong. Motion planning for an autonomous vehicle. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 529–533, 1988.

[103] P. Xavier. Private communication.

Biography

Hongyan Wang was born November 18, 1969, in Shannxi, China. After obtaining a B.S. in Computer Science from Peking University, Beijing, China, in June, 1991, she was accepted in the Ph.D. program by the Computer Science Department of Duke University. Her research interest is in the areas of algorithmic motion planning, computational geometry, and general algorithmic design. The following is a list of her publications:

1. John Reif and Hongyan Wang. Non-uniform discretization approximation for kinodynamic motion planning and its applications. To appear in *Proceedings of the 2nd Workshop on the Algorithmic Foundations of Robotics*, 1996.

2. Hongyan Wang and Pankaj K. Agarwal. Approximation algorithms for curvature-constrained shortest paths. In *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 409–418, 1996.

3. John Reif and Hongyan Wang. Social potential fields: A distributed behavioral control for autonomous robots. In *Proceedings of the 1st Workshop on the Algorithmic Foundations of Robotics*, pages 331–345, 1994.